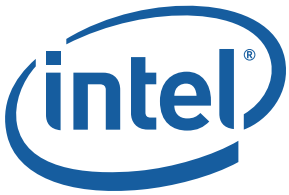

Intel® 64 and IA-32 Architectures Software Developer's Manual

Documentation Changes

March 2009

Notice: The Intel® 64 and IA-32 architectures may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in the specification updates.

Document Number: 252046-023

Contents

Revision History

Revision	Description	Date
-001	• Initial release	November 2002
-002	• Added 1-10 Documentation Changes. • Removed old Documentation Changes items that already have been incorporated in the published Software Developer's manual	December 2002
-003	• Added 9 -17 Documentation Changes. • Removed Documentation Change #6 - References to bits Gen and Len Deleted. • Removed Documentation Change #4 - VIF Information Added to CLI Discussion	February 2003
-004	• Removed Documentation changes 1-17. • Added Documentation changes 1-24.	June 2003
-005	• Removed Documentation Changes 1-24. • Added Documentation Changes 1-15.	September 2003
-006	• Added Documentation Changes 16- 34.	November 2003
-007	• Updated Documentation changes 14, 16, 17, and 28. • Added Documentation Changes 35-45.	January 2004
-008	• Removed Documentation Changes 1-45. • Added Documentation Changes 1-5.	March 2004
-009	• Added Documentation Changes 7-27.	May 2004
-010	• Removed Documentation Changes 1-27. • Added Documentation Changes 1.	August 2004
-011	• Added Documentation Changes 2-28.	November 2004
-012	• Removed Documentation Changes 1-28. • Added Documentation Changes 1-16.	March 2005
-013	• Updated title. • There are no Documentation Changes for this revision of the document.	July 2005
-014	• Added Documentation Changes 1-21.	September 2005
-015	• Removed Documentation Changes 1-21. • Added Documentation Changes 1-20.	March 9, 2006
-016	• Added Documentation changes 21-23.	March 27, 2006
-017	• Removed Documentation Changes 1-23. • Added Documentation Changes 1-36.	September 2006
-018	• Added Documentation Changes 37-42.	October 2006
-019	• Removed Documentation Changes 1-42. • Added Documentation Changes 1-19.	March 2007
-020	• Added Documentation Changes 20-27.	May 2007
-021	• Removed Documentation Changes 1-27. • Added Documentation Changes 1-6	November 2007
-022	• Removed Documentation Changes 1-6 • Added Documentation Changes 1-6	August 2008
-023	• Removed Documentation Changes 1-6 • Added Documentation Changes 1-21	March 2009

§

Preface

This document is an update to the specifications contained in the Affected Documents table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Affected Documents

Document Title	Document Number/Location
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture	253665
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M	253666
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z	253667
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide.	253668
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide	253669

Nomenclature

Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

Summary Tables of Changes

The following table indicates documentation changes which apply to the Intel® 64 and IA-32 architectures. This table uses the following notations:

Codes Used in Summary Tables

Change bar to left of table row indicates this erratum is either new or modified from the previous version of the document.

Documentation Changes

No.	DOCUMENTATION CHANGES
1	Pointers in 64-Bit Mode; updated figure 4-5 to include all four data pointers (Chapter 4, Vol 1)
2	Information Returned by CPUID Instruction (Chapter 3, Vol 2A)
3	Feature Information Returned in the ECX Register (Chapter 3, Vol 2A)
4	Error Handling (Chapter 9, Vol 3A)
5	APIC Timer (Chapter 9, Vol 3A)
6	Cache Management Instructions (Chapter 10, Vol 3A)
7	MTRR Feature Identification (Chapter 10, Vol 3A)
8	System-Management Range Register Interface (Chapter 10, Vol 3A)
9	Debug Status Register (DR6) (Chapter 18, Vol 3B)
10	Architectural Performance Monitoring Version 1 Facilities (Chapter 18, Vol 3B)
11	Real-mode Virtualization Updates to Chapter 19 (Chapter 19, Vol 3B)
12	Real-mode Virtualization Updates to Chapter 20 (Chapter 20, Vol 3B)
13	Real-mode Virtualization Updates to Chapter 21 (Chapter 21, Vol 3B)
14	Real-mode Virtualization Updates to Chapter 22 (Chapter 22, Vol 3B)
15	Real-mode Virtualization Updates to Chapter 23 (Chapter 23, Vol 3B)
16	Real-mode Virtualization Updates to Chapter 24 (Chapter 24, Vol 3B)
17	Real-mode Virtualization Updates to Chapter 25 (Chapter 25, Vol 3B)
18	Intel® Microarchitecture core and uncore events additions to Appendix A (Appendix A, Vol 3B)
19	Architectural PerfMon Updates and MSR Updates to Appendix B (Appendix B, Vol 3B)
20	Updated MC0 decoding definitions for processors, CMCI table correction to Appendix E (Appendix E, Vol 3B)
21	Real-mode Virtualization Updates to Appendix G (Appendix G, Vol 3B)

Documentation Changes

1. Pointers in 64-Bit Mode; updated figure 4-5 to include all four data pointers (Chapter 4, Vol 1)

Change bars show changes to Chapter 4 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture*.

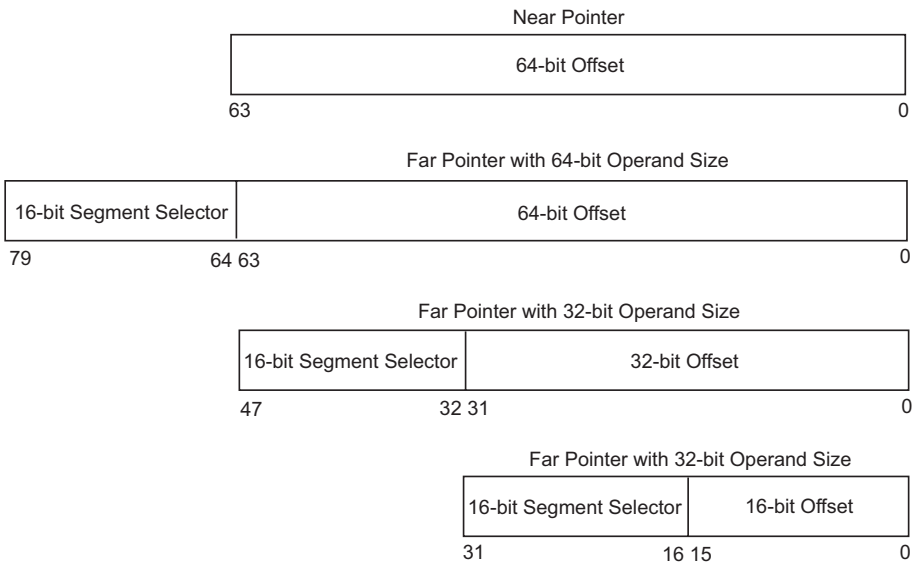

Figure 4-5. Pointers in 64-Bit Mode

2. Information Returned by CPUID Instruction (Chapter 3, Vol 2A)

Change bars show changes to Chapter 3 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M*.

Table 3-20. Information Returned by CPUID Instruction

Initial EAX Value	Information Provided about the Processor	
	Basic CPUID Information	
0H	EAX EBX ECX EDX	Maximum Input Value for Basic CPUID Information (see Table 3-21) "Genu" "ntel" "inel"
01H	EAX	Version Information: Type, Family, Model, and Stepping ID (see Figure 3-6)
	EBX	Bits 7-0: Brand Index Bits 15-8: CLFLUSH line size (Value ∗ 8 = cache line size in bytes) Bits 23-16: Maximum number of addressable IDs for logical processors in this physical package*. Bits 31-24: Initial APIC ID
	ECX EDX	Feature Information (see Figure 3-7 and Table 3-23) Feature Information (see Figure 3-8 and Table 3-24)
		NOTES: * The nearest power-of-2 integer that is not smaller than EBX[23:16] is the number of unique initial APIC IDs reserved for addressing different logical processors in a physical package.
02H	EAX EBX ECX EDX	Cache and TLB Information (see Table 3-25) Cache and TLB Information Cache and TLB Information Cache and TLB Information
03H	EAX EBX	Reserved. Reserved.
	ECX	Bits 00-31 of 96 bit processor serial number. (Available in Pentium III processor only; otherwise, the value in this register is reserved.)
	EDX	Bits 32-63 of 96 bit processor serial number. (Available in Pentium III processor only; otherwise, the value in this register is reserved.)
		NOTES: Processor serial number (PSN) is not supported in the Pentium 4 processor or later. On all models, use the PSN flag (returned using CPUID) to check for PSN support before accessing the feature.
		See AP-485, *Intel Processor Identification and the CPUID Instruction* (Order Number 241618) for more information on PSN.
	CPUID leaves > 3 < 80000000 are visible only when IA32_MISC_ENABLES.BOOT_NT4[bit 22] = 0 (default).	
	Deterministic Cache Parameters Leaf	
04H		**NOTES:** Leaf 04H output depends on the initial value in ECX. See also: "INPUT EAX = 4: Returns Deterministic Cache Parameters for each level on page 3-203.

Initial EAX Value	Information Provided about the Processor	
	EAX	Bits 4-0: Cache Type Field 0 = Null - No more caches 1 = Data Cache 2 = Instruction Cache 3 = Unified Cache 4-31 = Reserved Bits 7-5: Cache Level (starts at 1) Bits 8: Self Initializing cache level (does not need SW initialization) Bits 9: Fully Associative cache Bits 13-10: Reserved Bits 25-14: Maximum number of addressable IDs for logical processors sharing this cache*, ** Bits 31-26: Maximum number of addressable IDs for processor cores in the physical package*, ***, ****
	EBX	Bits 11-00: L = System Coherency Line Size* Bits 21-12: P = Physical Line partitions* Bits 31-22: W = Ways of associativity*
	ECX	Bits 31-00: S = Number of Sets*
	EDX	Bit 0: Write-Back Invalidate/Invalidate 0 = WBINVD/INVD from threads sharing this cache acts upon lower level caches for threads sharing this cache 1 = WBINVD/INVD is not guaranteed to act upon lower level caches of non-originating threads sharing this cache. Bit 1: Cache Inclusiveness 0 = Cache is not inclusive of lower cache levels. 1 = Cache is inclusive of lower cache levels. Bits 31-02: Reserved = 0
		NOTES: * Add one to the return value to get the result. ** The nearest power-of-2 integer that is not smaller than (1 + EAX[25:14]) is the number of unique initial APIC IDs reserved for addressing different logical processors sharing this cache *** The nearest power-of-2 integer that is not smaller than (1 + EAX[31:26]) is the number of unique Core_IDs reserved for addressing different processor cores in a physical package. Core ID is a subset of bits of the initial APIC ID. ****The returned value is constant for valid initial values in ECX. Valid ECX values start from 0.
	MONITOR/MWAIT Leaf	
5H	EAX	Bits 15-00: Smallest monitor-line size in bytes (default is processor's monitor granularity) Bits 31-16: Reserved = 0
	EBX	Bits 15-00: Largest monitor-line size in bytes (default is processor's monitor granularity) Bits 31-16: Reserved = 0

Initial EAX Value	Information Provided about the Processor	
	ECX	Bits 00: Enumeration of Monitor-Mwait extensions (beyond EAX and EBX registers) supported Bits 01: Supports treating interrupts as break-event for MWAIT, even when interrupts disabled Bits 31 - 02: Reserved
	EDX	Bits 03 - 00: Number of C0* sub C-states supported using MWait Bits 07 - 04: Number of C1* sub C-states supported using MWAIT Bits 11 - 08: Number of C2* sub C-states supported using MWAIT Bits 15 - 12: Number of C3* sub C-states supported using MWAIT Bits 19 - 16: Number of C4* sub C-states supported using MWAIT Bits 31 - 20: Reserved = 0 **NOTE:** * The definition of C0 through C4 states for MWAIT extension are processor-specific C-states, not ACPI C-states.
	Thermal and Power Management Leaf	
6H	EAX	Bit 00: Digital temperature sensor is supported if set Bit 01: Intel Turbo Boost Technology Available (see description of IA32_MISC_ENABLES[38]). Bit 02: ARAT. APIC-Timer-always-running feature is supported if set. Bits 31 - 03: Reserved
	EBX	Bits 03 - 00: Number of Interrupt Thresholds in Digital Thermal Sensor Bits 31 - 04: Reserved
	ECX	Bits 00: Hardware Coordination Feedback Capability (Presence of MCNT and ACNT MSRs). The capability to provide a measure of delivered processor performance (since last reset of the counters), as a percentage of expected processor performance at frequency specified in CPUID Brand String Bits 31 - 01: Reserved = 0
	EDX	Reserved = 0
	Direct Cache Access Information Leaf	
09H	EAX	Value of bits [31:0] of IA32_PLATFORM_DCA_CAP MSR (address 1F8H)
	EBX	Reserved
	ECX	Reserved
	EDX	Reserved
	Architectural Performance Monitoring Leaf	
0AH	EAX	Bits 07 - 00: Version ID of architectural performance monitoring Bits 15- 08: Number of general-purpose performance monitoring counter per logical processor Bits 23 - 16: Bit width of general-purpose, performance monitoring counter Bits 31 - 24: Length of EBX bit vector to enumerate architectural performance monitoring events

Initial EAX Value	Information Provided about the Processor	
	EBX	Bit 0: Core cycle event not available if 1 Bit 1: Instruction retired event not available if 1 Bit 2: Reference cycles event not available if 1 Bit 3: Last-level cache reference event not available if 1 Bit 4: Last-level cache misses event not available if 1 Bit 5: Branch instruction retired event not available if 1 Bit 6: Branch mispredict retired event not available if 1 Bits 31- 07: Reserved = 0
	ECX	Reserved = 0
	EDX	Bits 04 - 00: Number of fixed-function performance counters (if Version ID > 1) Bits 12- 05: Bit width of fixed-function performance counters (if Version ID > 1) Reserved = 0
	Extended Topology Enumeration Leaf	
0BH		**NOTES:** Most of Leaf 0BH output depends on the initial value in ECX. EDX output do not vary with initial value in ECX. ECX[7:0] output always reflect initial value in ECX. All other output value for an invalid initial value in ECX are 0. Leaf 0BH exists if EBX[15:0] is not zero.
	EAX	Bits 4-0: Number of bits to shift right on x2APIC ID to get a unique topology ID of the next level type*. All logical processors with the same next level ID share current level. Bits 31-5: Reserved.
	EBX	Bits 15 - 00: Number of logical processors at this level type. The number reflects configuration as shipped by Intel**. Bits 31- 16: Reserved.
	ECX	Bits 07 - 00: Level number. Same value in ECX input Bits 15 - 08: Level type***. Bits 31 - 16:: Reserved.
	EDX	Bits 31- 0: x2APIC ID the current logical processor.
		NOTES: * Software should use this field (EAX[4:0]) to enumerate processor topology of the system.

Initial EAX Value	Information Provided about the Processor
	** Software must not use EBX[15:0] to enumerate processor topology of the system. This value in this field (EBX[15:0]) is only intended for display/diagnostic purposes. The actual number of logical processors available to BIOS/OS/Applications may be different from the value of EBX[15:0], depending on software and platform hardware configurations. *** The value of the "level type" field is not related to level numbers in any way, higher "level type" values do not mean higher levels. Level type field has the following encoding: 0 : invalid 1 : SMT 2 : Core 3-255 : Reserved
	Processor Extended State Enumeration Main Leaf (EAX = 0DH, ECX = 0)
0DH	**NOTES:** Leaf 0DH main leaf (ECX = 0). EAX — Bits 31-0: Reports the valid bit fields of the lower 32 bits of the XFEATURE_ENABLED_MASK register (XCR0). If a bit is 0, the corresponding bit field in XCR0 is reserved.
	EBX — Bits 31-0: Maximum size (bytes) required by enabled features in XFEATURE_ENABLED_MASK (XCR0). May be different than ECX when features at the end of the save area are not enabled. ECX — Bit 31-0: Maximum size (bytes) of the XSAVE/XRSTOR save area required by all supported features in the processor, i.e all the valid bit fields in XFEATURE_ENABLED_MASK. This includes the size needed for the XSAVE.HEADER. EDX — Bit 31-0: Reports the valid bit fields of the upper 32 bits of the XFEATURE_ENABLED_MASK register (XCR0). If a bit is 0, the corresponding bit field in XCR0 is reserved
	Processor Extended State Enumeration Sub-leaf (EAX = 0DH, ECX = 1)
	EAX Reserved *EBX* Reserved *ECX* Reserved *EDX* Reserved
	Processor Extended State Enumeration Sub-leaves (EAX = 0DH, ECX = n, n > 1)
0DH	**NOTES:** Leaf 0DH output depends on the initial value in ECX. If ECX contains an invalid sub leaf index, EAX/EBX/ECX/EDX return 0.

Initial EAX Value	Information Provided about the Processor	
	EAX	Bits 31-0: The size in bytes of the save area for an extended state feature associated with a valid sub-leaf index, *n*. Each valid sub-leaf index maps to a valid bit in the XFEATURE_ENABLED_MASK register (XCR0) starting at bit position 2. This field reports 0 if the sub-leaf index, *n*, is invalid*.
	EBX	Bits 31-0: The offset in bytes of the save area from the beginning of the XSAVE/XRSTOR area. This field reports 0 if the sub-leaf index, *n*, is invalid*.
	ECX	This field reports 0 if the sub-leaf index, *n*, is invalid*; otherwise it is reserved.
	EDX	This field reports 0 if the sub-leaf index, *n*, is invalid*; otherwise it is reserved.
	Unimplemented CPUID Leaf Functions	
40000000H - 4FFFFFFFH		Invalid. No existing or future CPU will return processor identification or feature information if the initial EAX value is in the range 40000000H to 4FFFFFFFH. These leaves are not guaranteed to return 0.
	Extended Function CPUID Information	
80000000H	EAX	Maximum Input Value for Extended Function CPUID Information (see Table 3-21).
	EBX ECX EDX	Reserved Reserved Reserved
80000001H	EAX	Extended Processor Signature and Feature Bits.
	EBX	Reserved
	ECX	Bit 0: LAHF/SAHF available in 64-bit mode Bits 31-1 Reserved
	EDX	Bits 10-0: Reserved Bit 11: SYSCALL/SYSRET available (when in 64-bit mode) Bits 19-12: Reserved = 0 Bit 20: Execute Disable Bit available Bits 26-21: Reserved = 0 Bit 27: RDTSCP and IA32_TSC_AUX are available if 1 Bits 28: Reserved = 0 Bit 29: Intel® 64 Architecture available if 1 Bits 31-30: Reserved = 0
80000002H	EAX EBX ECX EDX	Processor Brand String Processor Brand String Continued Processor Brand String Continued Processor Brand String Continued

Initial EAX Value	Information Provided about the Processor	
80000003H	EAX EBX ECX EDX	Processor Brand String Continued Processor Brand String Continued Processor Brand String Continued Processor Brand String Continued
80000004H	EAX EBX ECX EDX	Processor Brand String Continued Processor Brand String Continued Processor Brand String Continued Processor Brand String Continued
80000005H	EAX EBX ECX EDX	Reserved = 0 Reserved = 0 Reserved = 0 Reserved = 0
80000006H	EAX EBX ECX EDX	Reserved = 0 Reserved = 0 Bits 7-0: Cache Line size in bytes Bits 15-12: L2 Associativity field * Bits 31-16: Cache size in 1K units Reserved = 0
		NOTES: * L2 associativity field encodings: 00H - Disabled 01H - Direct mapped 02H - 2-way 04H - 4-way 06H - 8-way 08H - 16-way 0FH - Fully associative
80000007H	EAX EBX ECX EDX	Reserved = 0 Reserved = 0 Reserved = 0 Bits 7-0: Reserved = 0 Bit 8: Invariant TSC available if 1 Bits 31-9: Reserved = 0
80000008H	EAX	Virtual/Physical Address size Bits 7-0: #Physical Address Bits* Bits 15-8: #Virtual Address Bits Bits 31-16: Reserved = 0
	EBX ECX EDX	Reserved = 0 Reserved = 0 Reserved = 0 **NOTES:** * If CPUID.80000008H:EAX[7:0] is supported, the maximum physical address number supported should come from this field.

3. Feature Information Returned in the ECX Register (Chapter 3, Vol 2A)

Change bars show changes to Chapter 3 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M*.

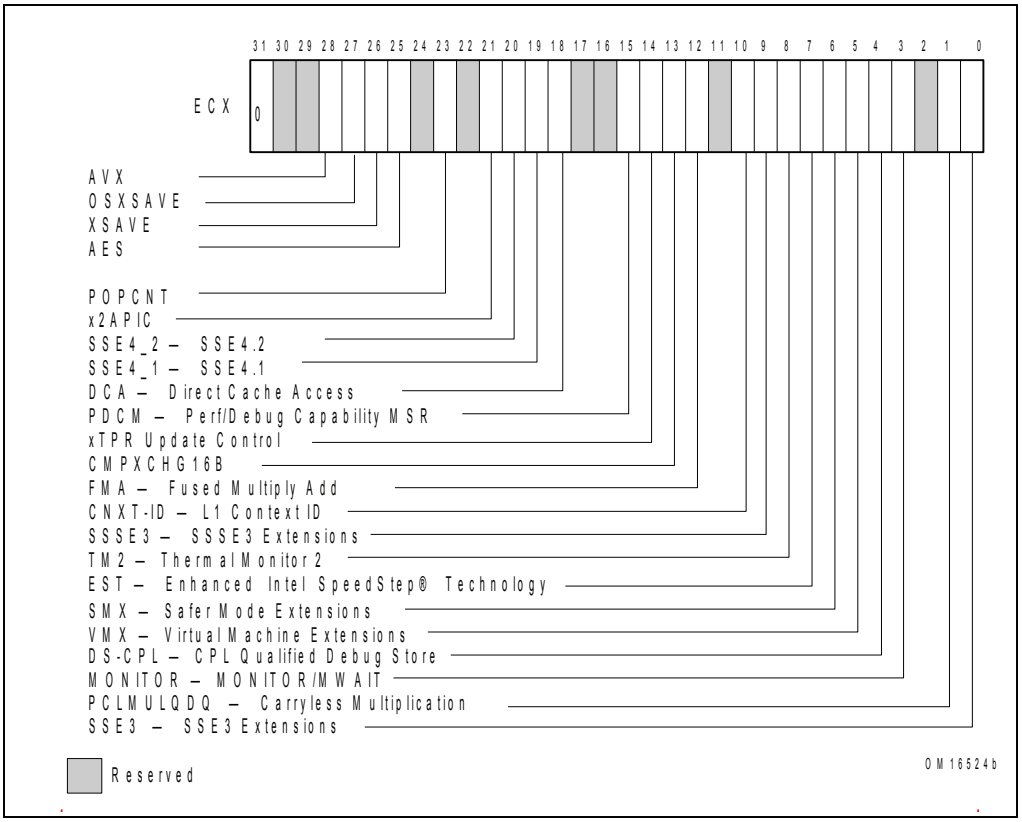

Figure 3-7. Feature Information Returned in the ECX Register

Table 3-23. Feature Information Returned in the ECX Register

Bit #	Mnemonic	Description
0	SSE3	**Streaming SIMD Extensions 3 (SSE3)**. A value of 1 indicates the processor supports this technology.
1	PCLMULQDQ	**PCLMULQDQ**. A value of 1 indicates the processor supports the PCLMULQDQ instruction
2	DTES64	**64-bit DS Area**. A value of 1 indicates the processor supports DS area using 64-bit layout
3	MONITOR	**MONITOR/MWAIT**. A value of 1 indicates the processor supports this feature.
4	DS-CPL	**CPL Qualified Debug Store**. A value of 1 indicates the processor supports the extensions to the Debug Store feature to allow for branch message storage qualified by CPL.
5	VMX	Virtual Machine Extensions. A value of 1 indicates that the processor supports this technology
6	SMX	Safer Mode Extensions. A value of 1 indicates that the processor supports this technology. See Chapter 6, "Safer Mode Extensions Reference".
7	EST	**Enhanced Intel SpeedStep® technology**. A value of 1 indicates that the processor supports this technology.
8	TM2	**Thermal Monitor 2**. A value of 1 indicates whether the processor supports this technology.
9	SSSE3	A value of 1 indicates the presence of the Supplemental Streaming SIMD Extensions 3 (SSSE3). A value of 0 indicates the instruction extensions are not present in the processor
10	CNXT-ID	**L1 Context ID**. A value of 1 indicates the L1 data cache mode can be set to either adaptive mode or shared mode. A value of 0 indicates this feature is not supported. See definition of the IA32_MISC_ENABLE MSR Bit 24 (L1 Data Cache Context Mode) for details.
11-12	Reserved	Reserved
13	CMPXCHG16B	**CMPXCHG16B Available**. A value of 1 indicates that the feature is available. See the "CMPXCHG8B/CMPXCHG16B—Compare and Exchange Bytes" section in this chapter for a description.
14	xTPR Update Control	**xTPR Update Control**. A value of 1 indicates that the processor supports changing IA32_MISC_ENABLES[bit 23].
15	PDCM	**Perfmon and Debug Capability**: A value of 1 indicates the processor supports the performance and debug feature indication MSR IA32_PERF_CAPABILITIES.
17 - 16	Reserved	Reserved
18	DCA	A value of 1 indicates the processor supports the ability to prefetch data from a memory mapped device.
19	SSE4.1	A value of 1 indicates that the processor supports SSE4.1.
20	SSE4.2	A value of 1 indicates that the processor supports SSE4.2.
21	x2APIC	A value of 1 indicates that the processor supports x2APIC feature.
22	MOVBE	A value of 1 indicates that the processor supports MOVBE instruction.

Bit #	Mnemonic	Description
23	POPCNT	A value of 1 indicates that the processor supports the POPCNT instruction.
24	Reserved	Reserved
25	AES	A value of 1 indicates that the processor supports the AES instruction extensions.
26	XSAVE	A value of 1 indicates that the processor supports the XSAVE/ XRSTOR processor extended states feature, the XSETBV/ XGETBV instructions, and the XFEATURE_ENABLED_MASK register (XCR0).
27	OSXSAVE	A value of 1 indicates that the OS has enabled XSETBV/XGETBV instructions to access the XFEATURE_ENABLED_MASK register (XCR0), and support for processor extended state management using XSAVE/XRSTOR.
30 - 28	Reserved	Reserved
31	Not Used	Always return 0

4. Error Handling (Chapter 9, Vol 3A)

Change bars show changes to Chapter 9 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

9.6.3 Error Handling

The local APIC provides an error status register (ESR) that it uses to record errors that it detects when handling interrupts (see Figure 9-13). An APIC error interrupt is generated when the local APIC sets one of the error bits in the ESR. The LVT error register allows selection of the interrupt vector to be delivered to the processor core when APIC error is detected. The LVT error register also provides a means of masking an APIC error interrupt.

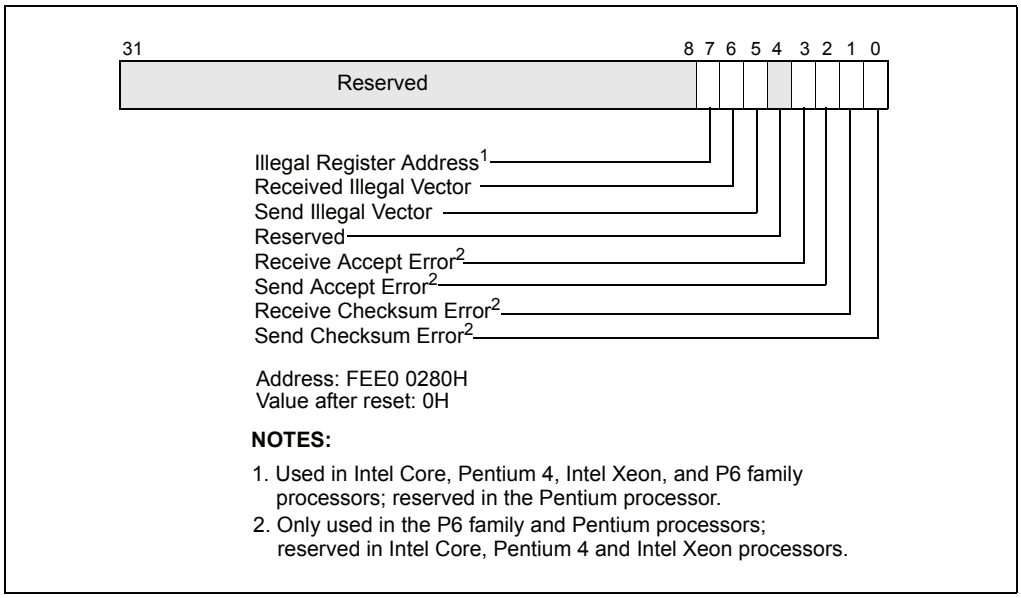

Figure 9-13. Error Status Register (ESR)

The ESR is a write/read register. A write (of any value) to the ESR must be done just prior to reading the ESR to update the register. This initial write causes the ESR contents to be updated with the latest error status. Back-to-back writes clear the ESR register. After an error bit is set in the register, it remains set until the register is cleared.

The functions of the ESR are listed in Table 2-1.

Table 2-1. ESR Flags

FLAG	Function
Send Checksum Error	(P6 family and Pentium processors only) Set when the local APIC detects a checksum error for a message that it sent on the APIC bus.
Receive Checksum Error	(P6 family and Pentium processors only) Set when the local APIC detects a checksum error for a message that it received on the APIC bus.
Send Accept Error	(P6 family and Pentium processors only) Set when the local APIC detects that a message it sent was not accepted by any APIC on the APIC bus.
Receive Accept Error	(P6 family and Pentium processors only) Set when the local APIC detects that the message it received was not accepted by any APIC on the APIC bus, including itself.
Send Illegal Vector	Set when the local APIC detects an illegal vector in the message that it is sending.
Receive Illegal Vector	Set when the local APIC detects an illegal vector in the message it received, including an illegal vector code in the local vector table interrupts or in a self-interrupt.

Table 2-1. ESR Flags

FLAG	Function
Send Checksum Error	(P6 family and Pentium processors only) Set when the local APIC detects a checksum error for a message that it sent on the APIC bus.
Receive Checksum Error	(P6 family and Pentium processors only) Set when the local APIC detects a checksum error for a message that it received on the APIC bus.
Illegal Reg. Address	(Intel Core, Intel Atom, Pentium 4, Intel Xeon, and P6 family processors only) Set when the processor is trying to access a register in the processor's local APIC register address space that is reserved (see Table 9-1). Addresses in one of the 0x10 byte regions marked reserved are illegal register addresses. The Local APIC Register Map is the address range of the APIC register base address (specified in the IA32_APIC_BASE MSR) plus 4 KBytes.

5. APIC Timer (Chapter 9, Vol 3A)

Change bars show changes to Chapter 9 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

9.6.4 APIC Timer

The local APIC unit contains a 32-bit programmable timer that is available to software to time events or operations. This timer is set up by programming four registers: the divide configuration register (see Figure 9-15), the initial-count and current-count registers (see Figure 9-16), and the LVT timer register (see Figure 9-12).

If CPUID.06H:EAX.ARAT[bit 2] is set, the processor's APIC timer runs at a constant rate regardless of P-state transitions and it continues to run at the same rate in deep C-states.

If CPUID.06H:EAX.ARAT[bit 2] = 0 or if CPUID 06H is not supported, the APIC timer may temporarily stop while the processor is in deep C-states or during transitions caused by Enhanced Intel SpeedStep® Technology.

6. Cache Management Instructions (Chapter 10, Vol 3A)

Change bars show changes to Chapter 10 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

10.5.5 Cache Management Instructions

The Intel 64 and IA-32 architectures provide several instructions for managing the L1, L2, and L3 caches. The INVD, WBINVD, and WBINVD instructions are system instructions that operate on the L1, L2, and L3 caches as a whole. The PREFETCH*h* and CLFLUSH instructions and the non-temporal move instructions (MOVNTI, MOVNTQ,

MOVNTDQ, MOVNTPS, and MOVNTPD), which were introduced in SSE/SSE2 extensions, offer more granular control over caching.

The INVD and WBINVD instructions are used to invalidate the contents of the L1, L2, and L3 caches. The INVD instruction invalidates all internal cache entries, then generates a special-function bus cycle that indicates that external caches also should be invalidated. The INVD instruction should be used with care. It does not force a write-back of modified cache lines; therefore, data stored in the caches and not written back to system memory will be lost. Unless there is a specific requirement or benefit to invalidating the caches without writing back the modified lines (such as, during testing or fault recovery where cache coherency with main memory is not a concern), software should use the WBINVD instruction.

The WBINVD instruction first writes back any modified lines in all the internal caches, then invalidates the contents of both the L1, L2, and L3 caches. It ensures that cache coherency with main memory is maintained regardless of the write policy in effect (that is, write-through or write-back). Following this operation, the WBINVD instruction generates one (P6 family processors) or two (Pentium and Intel486 processors) special-function bus cycles to indicate to external cache controllers that write-back of modified data followed by invalidation of external caches should occur. The amount of time or cycles for WBINVD to complete will vary due to the size of different cache hierarchies and other factors. As a consequence, the use of the WBINVD instruction can have an impact on interrupt/event response time.

The PREFETCH*h* instructions allow a program to suggest to the processor that a cache line from a specified location in system memory be prefetched into the cache hierarchy (see Section 10.8, “Explicit Caching”).

The CLFLUSH instruction allow selected cache lines to be flushed from memory. This instruction give a program the ability to explicitly free up cache space, when it is known that cached section of system memory will not be accessed in the near future.

The non-temporal move instructions (MOVNTI, MOVNTQ, MOVNTDQ, MOVNTPS, and MOVNTPD) allow data to be moved from the processor’s registers directly into system memory without being also written into the L1, L2, and/or L3 caches. These instructions can be used to prevent cache pollution when operating on data that is going to be modified only once before being stored back into system memory. These instructions operate on data in the general-purpose, MMX, and XMM registers.

7. MTRR Feature Identification (Chapter 10, Vol 3A)

Change bars show changes to Chapter 10 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

10.11.1 MTRR Feature Identification

The availability of the MTRR feature is model-specific. Software can determine if MTRRs are supported on a processor by executing the CPUID instruction and reading the state of the MTRR flag (bit 12) in the feature information register (EDX).

If the MTRR flag is set (indicating that the processor implements MTRRs), additional information about MTRRs can be obtained from the 64-bit IA32_MTRRCAP MSR (named MTRRcap MSR for the P6 family processors). The IA32_MTRRCAP MSR is a read-only MSR that can be read with the RDMSR instruction. Figure 10-5 shows the contents of the IA32_MTRRCAP MSR. The functions of the flags and field in this register are as follows:

- **VCNT (variable range registers count) field, bits 0 through 7** — Indicates the number of variable ranges implemented on the processor.
- **FIX (fixed range registers supported) flag, bit 8** — Fixed range MTRRs (IA32_MTRR_FIX64K_00000 through IA32_MTRR_FIX4K_0F8000) are supported when set; no fixed range registers are supported when clear.
- **WC (write combining) flag, bit 10** — The write-combining (WC) memory type is supported when set; the WC type is not supported when clear.
- **SMRR (System-Management Range Register) flag, bit 11** — The system-management range register (SMRR) interface is supported when bit 11 is set; the SMRR interface is not supported when clear.

Bit 9 and bits 11 through 63 in the IA32_MTRRCAP MSR are reserved. If software attempts to write to the IA32_MTRRCAP MSR, a general-protection exception (#GP) is generated.

Software must read IA32_MTRRCAP VCNT field to determine the number of variable MTRRs and query other feature bits in IA32_MTRRCAP to determine additional capabilities that are supported in a processor. For example, some processors may report a value of '8' in the VCNT field, other processors may report VCNT with different values.

8. System-Management Range Register Interface (Chapter 10, Vol 3A)

Change bars show changes to Chapter 10 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

10.11.2.4 System-Management Range Register Interface

If IA32_MTRRCAP[bit 11] is set, the processor supports the SMRR interface to restrict access to a specified memory address range used by system-management mode (SMM) software (see Section 25.4.2.1). If the SMRR interface is supported, SMM software is strongly encouraged to use it to protect the SMI code and data stored by SMI handler in the SMRAM region.

The system-management range registers consist of a pair of MSRs (see Figure 10-8). The IA32_SMRR_PHYSBASE MSR defines the base address for the SMRAM memory range and the memory type used to access it in SMM. The IA32_SMRR_PHYSMASK MSR contains a valid bit and a mask that determines the SMRAM address range protected by the SMRR interface. These MSRs may be written only in SMM; an attempt to write them outside of SMM causes a general-protection exception.[1]

Figure 10-8 shows flags and fields in these registers. The functions of these flags and fields are the following:

- **Type field, bits 0 through 7** — Specifies the memory type for the range (see Table 10-8 for the encoding of this field).
- **PhysBase field, bits 12 through 31** — Specifies the base address of the address range. The address must be less than 4 GBytes and is automatically aligned on a 4-KByte boundary.
- **PhysMask field, bits 12 through 31** — Specifies a mask that determines the range of the region being mapped, according to the following relationships:
  - Address_Within_Range AND PhysMask = PhysBase AND PhysMask
  - This value is extended by 12 bits at the low end to form the mask value. For more information: see Section 10.11.3, "Example Base and Mask Calculations."
- **V (valid) flag, bit 11** — Enables the register pair when set; disables register pair when clear.

Before attempting to access these SMRR registers, software must test bit 11 in the IA32_MTRRCAP register. If SMRR is not supported, reads from or writes to registers cause general-protection exceptions.

When the valid flag in the IA32_SMRR_PHYSMASK MSR is 1, accesses to the specified address range are treated as follows:

- If the logical processor is in SMM, accesses uses the memory type in the IA32_SMRR_PHYSBASE MSR.
- If the logical processor is not in SMM, write accesses are ignored and read accesses return 1's for all bits read. The uncacheable memory type (UC) is used in this case.

The above items apply even if the address range specified overlaps with a range specified by the MTRRs.

1. For some processor models, these MSRs can be accessed by RDMSR and WRMSR only if the SMRR interface has been enabled in the IA32_FEATURE_CONTROL MSR. See Appendix B.

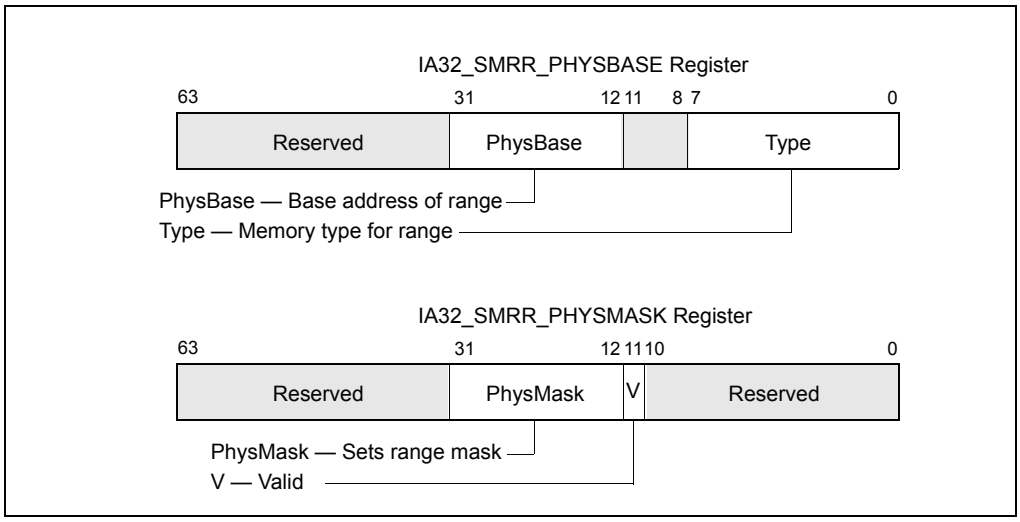

Figure 10-8 IA32_SMRR_PHYSBASE and IA32_SMRR_PHYSMASK SMRR Pair

9. Debug Status Register (DR6) (Chapter 18, Vol 3B)

Change bars show changes to Chapter 18 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

18.2.3 Debug Status Register (DR6)

The debug status register (DR6) reports debug conditions that were sampled at the time the last debug exception was generated (see Figure 18-1). Updates to this register only occur when an exception is generated. The flags in this register show the following information:

- **B0 through B3 (breakpoint condition detected) flags (bits 0 through 3) —** Indicates (when set) that its associated breakpoint condition was met when a debug exception was generated. These flags are set if the condition described for each breakpoint by the LEN*n*, and R/W*n* flags in debug control register DR7 is true. They may or may not be set if the breakpoint is not enabled by the Ln or the Gn flags in register DR7. Therefore on a #DB, a debug handler should check only those B0-B3 bits which correspond to an enabled breakpoint.
- **BD (debug register access detected) flag (bit 13) —** Indicates that the next instruction in the instruction stream accesses one of the debug registers (DR0 through DR7). This flag is enabled when the GD (general detect) flag in debug control register DR7 is set. See Section 18.2.4, "Debug Control Register (DR7)," for further explanation of the purpose of this flag.
- **BS (single step) flag (bit 14) —** Indicates (when set) that the debug exception was triggered by the single-step execution mode (enabled with the TF flag in the EFLAGS register). The single-step mode is the highest-priority debug exception. When the BS flag is set, any of the other debug status bits also may be set.
- **BT (task switch) flag (bit 15) —** Indicates (when set) that the debug exception resulted from a task switch where the T flag (debug trap flag) in the TSS of the target

task was set. See Section 6.2.1, “Task-State Segment (TSS),” for the format of a TSS. There is no flag in debug control register DR7 to enable or disable this exception; the T flag of the TSS is the only enabling flag.

Certain debug exceptions may clear bits 0-3. The remaining contents of the DR6 register are never cleared by the processor. To avoid confusion in identifying debug exceptions, debug handlers should clear the register before returning to the interrupted task.

10. Architectural Performance Monitoring Version 1 Facilities (Chapter 18, Vol 3B)

Change bars show changes to Chapter 18 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

18.13.1.1 Architectural Performance Monitoring Version 1 Facilities

Architectural performance monitoring facilities include a set of performance monitoring counters and performance event select registers. These MSRs have the following properties:

- IA32_PMCx MSRs start at address 0C1H and occupy a contiguous block of MSR address space; the number of MSRs per logical processor is reported using CPUID.0AH:EAX[15:8].
- IA32_PERFEVTSELx MSRs start at address 186H and occupy a contiguous block of MSR address space. Each performance event select register is paired with a corresponding performance counter in the 0C1H address block.
- The bit width of an IA32_PMCx MSR is reported using the CPUID.0AH:EAX[23:16]. Bits beyond the width of the programmable counter are undefined, and are ignored when written to. On write operations, the lower-order 32 bits of each MSR may be written with any value, and the high-order bits are sign-extended according to the value of bit 31. The bit width for read operations is reported using CPUID.
- Bit field layout of IA32_PERFEVTSELx MSRs is defined architecturally.

See Figure 18-13 for the bit field layout of IA32_PERFEVTSELx MSRs. The bit fields are:

- **Event select field (bits 0 through 7) —** Selects the event logic unit used to detect microarchitectural conditions (see Table 18-10, for a list of architectural events and their 8-bit codes). The set of values for this field is defined architecturally; each value corresponds to an event logic unit for use with an architectural performance event. The number of architectural events is queried using CPUID.0AH:EAX. A processor may support only a subset of pre-defined values.

11. Real-mode Virtualization Updates to Chapter 19 (Chapter 19, Vol 3B)

Change bars show changes to Chapter 19 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

19.8 RESTRICTIONS ON VMX OPERATION

VMX operation places restrictions on processor operation. These are detailed below:

- In VMX operation, processors may fix certain bits in CR0 and CR4 to specific values and not support other values. VMXON fails if any of these bits contains an unsupported value (see "VMXON—Enter VMX Operation" in Chapter 5 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B*). Any attempt to set one of these bits to an unsupported value while in VMX operation (including VMX root operation) using any of the CLTS, LMSW, or MOV CR instructions causes a general-protection exception. VM entry or VM exit cannot set any of these bits to an unsupported value.[1]

NOTES

The first processors to support VMX operation require that the following bits be 1 in VMX operation: CR0.PE, CR0.NE, CR0.PG, and CR4.VMXE. The restrictions on CR0.PE and CR0.PG imply that VMX operation is supported only in paged protected mode (including IA-32e mode). Therefore, guest software cannot be run in unpaged protected mode or in real-address mode. See Section 26.2, "Supporting Processor Operating Modes in Guest Environments," for a discussion of how a VMM might support guest software that expects to run in unpaged protected mode or in real-address mode.

Later processors support a VM-execution control called "unrestricted guest" (see Section 20.6.2). If this control is 1, CR0.PE and CR0.PG may be 0 in VMX non-root operation (even if the capability MSR IA32_VMX_CR0_FIXED1 reports otherwise).[2] Such processors allow guest software to run in unpaged protected mode or in real-address mode.

VMXON fails if a logical processor is in A20M mode (see "VMXON—Enter VMX Operation" in Chapter 6 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B*). Once the processor is in VMX operation, A20M

1. Software should consult the VMX capability MSRs IA32_VMX_CR0_FIXED0 and IA32_VMX_CR0_FIXED1 to determine how bits in CR0 are set. (see Appendix G.7). For CR4, software should consult the VMX capability MSRs IA32_VMX_CR4_FIXED0 and IA32_VMX_CR4_FIXED1 (see Appendix G.8).
2. "Unrestricted guest" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "unrestricted guest" VM-execution control were 0. See Section 20.6.2.

12. Real-mode Virtualization Updates to Chapter 20 (Chapter 20, Vol 3B)

Change bars show changes to Chapter 20 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

20.4.1 Guest Register State

The following fields in the guest-state area correspond to processor registers:

- Control registers CR0, CR3, and CR4 (64 bits each; 32 bits on processors that do not support Intel 64 architecture).
- Debug register DR7 (64 bits; 32 bits on processors that do not support Intel 64 architecture).
- RSP, RIP, and RFLAGS (64 bits each; 32 bits on processors that do not support Intel 64 architecture).[1]
- The following fields for each of the registers CS, SS, DS, ES, FS, GS, LDTR, and TR:
  - Selector (16 bits).
  - Base address (64 bits; 32 bits on processors that do not support Intel 64 architecture). The base-address fields for CS, SS, DS, and ES have only 32 architecturally-defined bits; nevertheless, the corresponding VMCS fields have 64 bits on processors that support Intel 64 architecture.
  - Segment limit (32 bits). The limit field is always a measure in bytes.
  - Access rights (32 bits). The format of this field is given in Table 20-2 and detailed as follows:
    - The low 16 bits correspond to bits 23:8 of the upper 32 bits of a 64-bit segment descriptor. While bits 19:16 of code-segment and data-segment descriptors correspond to the upper 4 bits of the segment limit, the corresponding bits (bits 11:8) are reserved in this VMCS field.
    - Bit 16 indicates an **unusable segment**. Attempts to use such a segment fault except in 64-bit mode. In general, a segment register is unusable if it has been loaded with a null selector.[2]
    - Bits 31:17 are reserved.

1. This chapter uses the notation RAX, RIP, RSP, RFLAGS, etc. for processor registers because most processors that support VMX operation also support Intel 64 architecture. For processors that do not support Intel 64 architecture, this notation refers to the 32-bit forms of those registers (EAX, EIP, ESP, EFLAGS, etc.). In a few places, notation such as EAX is used to refer specifically to lower 32 bits of the indicated register.
2. There are a few exceptions to this statement. For example, a segment with a non-null selector may be unusable following a task switch that fails after its commit point; see "Interrupt 10—Invalid TSS Exception (#TS)" in Section 5.14, "Exception and Interrupt Handling in 64-bit Mode," of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*. In contrast, the TR register is usable after processor reset despite having a null selector; see Table 9-1 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*.

Table 20-2 Format of Access Rights

Bit Position(s)	Field
3:0	Segment type
4	S — Descriptor type (0 = system; 1 = code or data)
6:5	DPL — Descriptor privilege level
7	P — Segment present
11:8	Reserved
12	AVL — Available for use by system software
13	Reserved (except for CS) L — 64-bit mode active (for CS only)
14	D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
15	G — Granularity
16	Segment unusable (0 = usable; 1 = unusable)
31:17	Reserved

The base address, segment limit, and access rights compose the "hidden" part (or "descriptor cache") of each segment register. These data are included in the VMCS because it is possible for a segment register's descriptor cache to be inconsistent with the segment descriptor in memory (in the GDT or the LDT) referenced by the segment register's selector.

The value of the DPL field for SS is always equal to the logical processor's current privilege level (CPL).[1]

1. In protected mode, CPL is also associated with the RPL field in the CS selector. However, the RPL fields are not meaningful in real-address mode or in virtual-8086 mode.

20.6.1 Pin-Based VM-Execution Controls

The pin-based VM-execution controls constitute a 32-bit vector that governs the handling of asynchronous events (for example: interrupts).[1] Table 20-5 lists the controls supported. See Chapter 21 for how these controls affect processor behavior in VMX non-root operation.

Table 20-5 Definitions of Pin-Based VM-Execution Controls

Bit Position(s)	Name	Description
0	External-interrupt exiting	If this control is 1, external interrupts cause VM exits. Otherwise, they are delivered normally through the guest interrupt-descriptor table (IDT). If this control is 1, the value of RFLAGS.IF does not affect interrupt blocking.
3	NMI exiting	If this control is 1, non-maskable interrupts (NMIs) cause VM exits. Otherwise, they are delivered normally using descriptor 2 of the IDT. This control also determines interactions between IRET and blocking by NMI (see Section 21.4).
5	Virtual NMIs	If this control is 1, NMIs are never blocked and the "blocking by NMI" bit (bit 3) in the interruptibility-state field indicates "virtual-NMI blocking" (see Table 20-3). This control also interacts with the "NMI-window exiting" VM-execution control (see Section 20.6.2). This control can be set only if the "NMI exiting" VM-execution control (above) is 1.
6	Activate VMX-preemption timer	If this control is 1, the VMX-preemption timer counts down in VMX non-root operation; see Section 21.7.1. A VM exit occurs when the timer counts down to zero; see Section 21.3.

All other bits in this field are reserved, some to 0 and some to 1. Software should consult the VMX capability MSRs IA32_VMX_PINBASED_CTLS and IA32_VMX_TRUE_PINBASED_CTLS (see Appendix G.3.1) to determine how to set reserved bits. Failure to set reserved bits properly causes subsequent VM entries to fail (see Section 22.2).

The first processors to support the virtual-machine extensions supported only the 1-settings of bits 1, 2, and 4. The VMX capability MSR IA32_VMX_PINBASED_CTLS will always report that these bits must be 1. Logical processors that support the 0-settings of any of these bits will support the VMX capability MSR IA32_VMX_TRUE_PINBASED_CTLS MSR, and software should consult this MSR to discover support for the 0-settings of these bits. Software that is not aware of the functionality of any one of these bits should set that bit to 1.

20.6.2 Processor-Based VM-Execution Controls

The processor-based VM-execution controls constitute two 32-bit vectors that govern the handling of synchronous events, mainly those caused by the execution of specific

1. Some asynchronous events cause VM exits regardless of the settings of the pin-based VM-execution controls (see Section 21.3).

instructions.[1] These are the **primary processor-based VM-execution controls** and the **secondary processor-based VM-execution controls**.

Table 20-6 lists the primary processor-based VM-execution controls. See Chapter 21 for more details of how these controls affect processor behavior in VMX non-root operation.

Table 20-6 Definitions of Primary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
2	Interrupt-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if RFLAGS.IF = 1 and there are no other blocking of interrupts (see Section 20.4.2).
3	Use TSC offsetting	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC offset field (see Section 20.6.5 and Section 21.4).
7	HLT exiting	This control determines whether executions of HLT cause VM exits.
9	INVLPG exiting	This determines whether executions of INVLPG cause VM exits.
10	MWAIT exiting	This control determines whether executions of MWAIT cause VM exits.
11	RDPMC exiting	This control determines whether executions of RDPMC cause VM exits.
12	RDTSC exiting	This control determines whether executions of RDTSC and RDTSCP cause VM exits.
15	CR3-load exiting	In conjunction with the CR3-target controls (see Section 20.6.7), this control determines whether executions of MOV to CR3 cause VM exits. See Section 21.1.3. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
16	CR3-store exiting	This control determines whether executions of MOV from CR3 cause VM exits. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
19	CR8-load exiting	This control determines whether executions of MOV to CR8 cause VM exits. This control must be 0 on processors that do not support Intel 64 architecture.
20	CR8-store exiting	This control determines whether executions of MOV from CR8 cause VM exits. This control must be 0 on processors that do not support Intel 64 architecture.

1. Some instructions cause VM exits regardless of the settings of the processor-based VM-execution controls (see Section 21.1.2), as do task switches (see Section 21.3).

Table 20-6 Definitions of Primary Processor-Based VM-Execution Controls (Continued)

Bit Position(s)	Name	Description
21	Use TPR shadow	Setting this control to 1 activates the TPR shadow, which is maintained in a page of memory addressed by the virtual-APIC address. See Section 21.4. This control must be 0 on processors that do not support Intel 64 architecture.
22	NMI-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if there is no virtual-NMI blocking (see Section 20.4.2). This control can be set only if the "virtual NMIs" VM-execution control (see Section 20.6.1) is 1.
23	MOV-DR exiting	This control determines whether executions of MOV DR cause VM exits.
24	Unconditional I/O exiting	This control determines whether executions of I/O instructions (IN, INS/INSB/INSW/INSD, OUT, and OUTS/OUTSB/OUTSW/OUTSD) cause VM exits. This control is ignored if the "use I/O bitmaps" control is 1.
25	Use I/O bitmaps	This control determines whether I/O bitmaps are used to restrict executions of I/O instructions (see Section 20.6.4 and Section 21.1.3). For this control, "0" means "do not use I/O bitmaps" and "1" means "use I/O bitmaps." If the I/O bitmaps are used, the setting of the "unconditional I/O exiting" control is ignored.
27	Monitor trap flag	If this control is 1, the monitor trap flag debugging feature is enabled. See Section 21.7.2.
28	Use MSR bitmaps	This control determines whether MSR bitmaps are used to control execution of the RDMSR and WRMSR instructions (see Section 20.6.9 and Section 21.1.3). For this control, "0" means "do not use MSR bitmaps" and "1" means "use MSR bitmaps." If the MSR bitmaps are not used, all executions of the RDMSR and WRMSR instructions cause VM exits. Not all processors support the 1-setting of this control. Software may consult the VMX capability MSR IA32_VMX_PROCBASED_CTLS (see Appendix G.3) to determine whether that setting is supported.
29	MONITOR exiting	This control determines whether executions of MONITOR cause VM exits.
30	PAUSE exiting	This control determines whether executions of PAUSE cause VM exits.
31	Activate secondary controls	This control determines whether the secondary processor-based VM-execution controls are used. If this control is 0, the logical processor operates as if all the secondary processor-based VM-execution controls were also 0.

All other bits in this field are reserved, some to 0 and some to 1. Software should consult the VMX capability MSRs IA32_VMX_PROCBASED_CTLS and IA32_VMX_TRUE_PROCBASED_CTLS (see Appendix G.3.2) to determine how to set

reserved bits. Failure to set reserved bits properly causes subsequent VM entries to fail (see Section 22.2).

The first processors to support the virtual-machine extensions supported only the 1-settings of bits 1, 4–6, 8, 13–16, and 26. The VMX capability MSR IA32_VMX_PROCBASED_CTLS will always report that these bits must be 1. Logical processors that support the 0-settings of any of these bits will support the VMX capability MSR IA32_VMX_TRUE_PROCBASED_CTLS MSR, and software should consult this MSR to discover support for the 0-settings of these bits. Software that is not aware of the functionality of any one of these bits should set that bit to 1.

Bit 31 of the primary processor-based VM-execution controls determines whether the secondary processor-based VM-execution controls are used. If that bit is 0, VM entry and VMX non-root operation function as if all the secondary processor-based VM-execution controls were 0. Processors that support only the 0-setting of bit 31 of the primary processor-based VM-execution controls do not support the secondary processor-based VM-execution controls.

Table 20-7 lists the secondary processor-based VM-execution controls. See Chapter 21 for more details of how these controls affect processor behavior in VMX non-root operation.

Table 20-7 Definitions of Secondary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
0	Virtualize APIC accesses	If this control is 1, a VM exit occurs on any attempt to access data on the page with the APIC-access address. See Section 21.2.
1	Enable EPT	If this control is 1, extended page tables (EPT) are enabled. See Chapter 24.
2	Descriptor-table exiting	This control determines whether executions of LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, and STR cause VM exits.
3	Enable RDTSCP	If this control is 0, any execution of RDTSCP causes and invalid-opcode exception (#UD).
4	Virtualize x2APIC mode	Setting this control to 1 causes RDMSR and WRMSR to MSR 808H to use the TPR shadow, which is maintained on the virtual-APIC page. See Section 21.4.
5	Enable VPID	If this control is 1, cached translations of linear addresses are associated with a virtual-processor identifier (VPID). See Chapter 24.1.
6	WBINVD exiting	This control determines whether executions of WBINVD cause VM exits.
7	Unrestricted guest	This control determines whether guest software may run in unpaged protected mode or in real-address mode.

20.6.8 Controls for APIC Accesses

There are three mechanisms by which software accesses registers of the logical processor's local APIC:

- If the local APIC is in xAPIC mode, it can perform memory-mapped accesses to addresses in the 4-KByte page referenced by the physical address in the IA32_APIC_BASE MSR (see Section 9.4.4, "Local APIC Status and Location" in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A* and *Intel® 64 Architecture Processor Topology Enumeration*).[1]
- If the local APIC is in x2APIC mode, it can accesses the local APIC's registers using the RDMSR and WRMSR instructions (see *Intel® 64 Architecture Processor Topology Enumeration*).
- In 64-bit mode, it can access the local APIC's task-priority register (TPR) using the MOV CR8 instruction.

There are three processor-based VM-execution controls (see Section 20.6.2) that control such accesses. There are "use TPR shadow", "virtualize APIC accesses", and "virtualize x2APIC mode". These controls interact with the following fields:

- **APIC-access address** (64 bits). This field is the physical address of the 4-KByte **APIC-access page**. If the "virtualize APIC accesses" VM-execution control is 1, operations that access this page may cause VM exits. See Section 21.2 and Section 21.5.

  The APIC-access address exists only on processors that support the 1-setting of the "virtualize APIC accesses" VM-execution control.

1. If the local APIC does not support x2APIC mode, it is always in xAPIC mode.

20.7.1 VM-Exit Controls

The VM-exit controls constitute a 32-bit vector that governs the basic operation of VM exits. Table 20-9 lists the controls supported. See Chapter 23 for complete details of how these controls affect VM exits.

Table 20-9 Definitions of VM-Exit Controls

Bit Position(s)	Name	Description
2	Save debug controls	This control determines whether DR7 and the IA32_DEBUGCTL MSR are saved on VM exit. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
9	Host address-space size	On processors that support Intel 64 architecture, this control determines whether a logical processor is in 64-bit mode after the next VM exit. Its value is loaded into CS.L, IA32_EFER.LME, and IA32_EFER.LMA on every VM exit.[1] This control must be 0 on processors that do not support Intel 64 architecture.
12	Load IA32_PERF_GLOBAL_CTRL	This control determines whether the IA32_PERF_GLOBAL_CTRL MSR is loaded on VM exit.
15	Acknowledge interrupt on exit	This control affects VM exits due to external interrupts: ▪ If such a VM exit occurs and this control is 1, the logical processor acknowledges the interrupt controller, acquiring the interrupt's vector. The vector is stored in the VM-exit interruption-information field, which is marked valid. ▪ If such a VM exit occurs and this control is 0, the interrupt is not acknowledged and the VM-exit interruption-information field is marked invalid.
18	Save IA32_PAT	This control determines whether the IA32_PAT MSR is saved on VM exit.
19	Load IA32_PAT	This control determines whether the IA32_PAT MSR is loaded on VM exit.
20	Save IA32_EFER	This control determines whether the IA32_EFER MSR is saved on VM exit.
21	Load IA32_EFER	This control determines whether the IA32_EFER MSR is loaded on VM exit.
22	Save VMX-preemption timer value	This control determines whether the value of the VMX-preemption timer is saved on VM exit.

NOTES:
1. Since Intel 64 architecture specifies that IA32_EFER.LMA is always set to the logical-AND of CR0.PG and IA32_EFER.LME, and since CR0.PG is always 1 in VMX operation, IA32_EFER.LMA is always identical to IA32_EFER.LME in VMX operation.

All other bits in this field are reserved, some to 0 and some to 1. Software should consult the VMX capability MSRs IA32_VMX_EXIT_CTLS and IA32_VMX_TRUE_EXIT_CTLS (see Appendix G.4) to determine how it should set the reserved bits. Failure to set reserved bits properly causes subsequent VM entries to fail (see Section 22.2).

The first processors to support the virtual-machine extensions supported only the 1-settings of bits 0–8, 10, 11, 13, 14, 16, and 17. The VMX capability MSR IA32_VMX_EXIT_CTLS always reports that these bits must be 1. Logical processors that support the 0-settings of any of these bits will support the VMX capability MSR IA32_VMX_TRUE_EXIT_CTLS MSR, and software should consult this MSR to discover support for the 0-settings of these bits. Software that is not aware of the functionality of any one of these bits should set that bit to 1.

20.8.1 VM-Entry Controls

The VM-entry controls constitute a 32-bit vector that governs the basic operation of VM entries. Table 20-11 lists the controls supported. See Chapter 22 for how these controls affect VM entries.

Table 20-11 Definitions of VM-Entry Controls

Bit Position(s)	Name	Description
2	Load debug controls	This control determines whether DR7 and the IA32_DEBUGCTL MSR are loaded on VM exit. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
9	IA-32e mode guest	On processors that support Intel 64 architecture, this control determines whether the logical processor is in IA-32e mode after VM entry. Its value is loaded into IA32_EFER.LMA as part of VM entry.[1] This control must be 0 on processors that do not support Intel 64 architecture.
10	Entry to SMM	This control determines whether the logical processor is in system-management mode (SMM) after VM entry. This control must be 0 for any VM entry from outside SMM.
11	Deactivate dual-monitor treatment	If set to 1, the default treatment of SMIs and SMM is in effect after the VM entry (see Section 25.15.7). This control must be 0 for any VM entry from outside SMM.
13	Load IA32_PERF_GLOBAL_CTRL	This control determines whether the IA32_PERF_GLOBAL_CTRL MSR is loaded on VM entry.
14	Load IA32_PAT	This control determines whether the IA32_PAT MSR is loaded on VM entry.
15	Load IA32_EFER	This control determines whether the IA32_EFER MSR is loaded on VM entry.

NOTES:

1. Bit 5 of the IA32_VMX_MISC MSR is read as 1 on any logical processor that supports the 1-setting of the "unrestricted guest" VM-execution control. If it is read as 1, every VM exit stores the value of IA32_EFER.LMA into the "IA-32e mode guest" VM-entry control (see Section 23.2).

20.8.3 VM-Entry Controls for Event Injection

VM entry can be configured to conclude by delivering an event through the IDT (after all guest state and MSRs have been loaded). This process is called **event injection** and is controlled by the following three VM-entry control fields:

- **VM-entry interruption-information field** (32 bits). This field provides details about the event to be injected. Table 20-12 describes the field.

Table 20-12 Format of the VM-Entry Interruption-Information Field

Bit Position(s)	Content
7:0	Vector of interrupt or exception
10:8	Interruption type: 0: External interrupt 1: Reserved 2: Non-maskable interrupt (NMI) 3: Hardware exception 4: Software interrupt 5: Privileged software exception 6: Software exception 7: Other event
11	Deliver error code (0 = do not deliver; 1 = deliver)
30:12	Reserved
31	Valid

  - The **vector** (bits 7:0) determines which entry in the IDT is used or which other event is injected.
  - The **interruption type** (bits 10:8) determines details of how the injection is performed. In general, a VMM should use the type **hardware exception** for all exceptions other than breakpoint exceptions (#BP; generated by INT3) and overflow exceptions (#OF; generated by INTO); it should use the type **software exception** for #BP and #OF. The type **other event** is used for injection of events that are not delivered through the IDT.
  - For exceptions, the **deliver-error-code bit** (bit 11) determines whether delivery pushes an error code on the guest stack.
  - VM entry injects an event if and only if the **valid bit** (bit 31) is 1. The valid bit in this field is cleared on every VM exit (see Section 23.2).

20.10.3 Software Access to Related Structures

In addition to data in the VMCS region itself, VMX non-root operation can be controlled by data structures that are referenced by pointers in a VMCS (for example, the I/O bitmaps). While the pointers to these data structures are parts of the VMCS, the data structures themselves are not. They are not accessible using VMREAD and VMWRITE but by ordinary memory writes.

Software should ensure that each such data structure is modified only when no logical processor with a current VMCS that references it is in VMX non-root operation. Doing otherwise may lead to unpredictable behavior (including behaviors identified in Section 20.10.1).

13. Real-mode Virtualization Updates to Chapter 21 (Chapter 21, Vol 3B)

Change bars show changes to Chapter 21 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

21.1.3 Instructions That Cause VM Exits Conditionally

Certain instructions cause VM exits in VMX non-root operation depending on the setting of the VM-execution controls. The following instructions can cause "fault-like" VM exits based on the conditions described:

- **CLTS**. The CLTS instruction causes a VM exit if the bits in position 3 (corresponding to CR0.TS) are set in both the CR0 guest/host mask and the CR0 read shadow.
- **HLT.** The HLT instruction causes a VM exit if the "HLT exiting" VM-execution control is 1.
- **IN, INS/INSB/INSW/INSD, OUT, OUTS/OUTSB/OUTSW/OUTSD.** The behavior of each of these instructions is determined by the settings of the "unconditional I/O exiting" and "use I/O bitmaps" VM-execution controls:
  - If both controls are 0, the instruction executes normally.
  - If the "unconditional I/O exiting" VM-execution control is 1 and the "use I/O bitmaps" VM-execution control is 0, the instruction causes a VM exit.
  - If the "use I/O bitmaps" VM-execution control is 1, the instruction causes a VM exit if it attempts to access an I/O port corresponding to a bit set to 1 in the appropriate I/O bitmap (see Section 20.6.4). If an I/O operation "wraps around" the 16-bit I/O-port space (accesses ports FFFFH and 0000H), the I/O instruction causes a VM exit (the "unconditional I/O exiting" VM-execution control is ignored if the "use I/O bitmaps" VM-execution control is 1).

  See Section 21.1.1 for information regarding the priority of VM exits relative to faults that may be caused by the INS and OUTS instructions.
- **INLVPG.** The INLVPG instruction causes a VM exit if the "INLVPG exiting" VM-execution control is 1.
- **LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, STR.** These instructions cause VM exits if the "descriptor-table exiting" VM-execution control is 1.[1]
- **LMSW.** In general, the LMSW instruction causes a VM exit if it would write, for any bit set in the low 4 bits of the CR0 guest/host mask, a value different than the corresponding bit in the CR0 read shadow. LMSW never clears bit 0 of CR0 (CR0.PE); thus, LMSW causes a VM exit if either of the following are true:
  - The bits in position 0 (corresponding to CR0.PE) are set in both the CR0 guest/mask and the source operand, and the bit in position 0 is clear in the CR0 read shadow.
  - For any bit position in the range 3:1, the bit in that position is set in the CR0 guest/mask and the values of the corresponding bits in the source operand and the CR0 read shadow differ.

1. "Descriptor-table exiting" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "descriptor-table exiting" VM-execution control were 0. See Section 20.6.2.

- **MONITOR.** The MONITOR instruction causes a VM exit if the "MONITOR exiting" VM-execution control is 1.
- **MOV from CR3.** The MOV from CR3 instruction causes a VM exit if the "CR3-store exiting" VM-execution control is 1. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
- **MOV from CR8.** The MOV from CR8 instruction (which can be executed only in 64-bit mode) causes a VM exit if the "CR8-store exiting" VM-execution control is 1. If this control is 0, the behavior of the MOV from CR8 instruction is modified if the "use TPR shadow" VM-execution control is 1 (see Section 21.4).
- **MOV to CR0.** The MOV to CR0 instruction causes a VM exit unless the value of its source operand matches, for the position of each bit set in the CR0 guest/host mask, the corresponding bit in the CR0 read shadow. (If every bit is clear in the CR0 guest/host mask, MOV to CR0 cannot cause a VM exit.)
- **MOV to CR3.** The MOV to CR3 instruction causes a VM exit unless the "CR3-load exiting" VM-execution control is 0 or the value of its source operand is equal to one of the CR3-target values specified in the VMCS. If the CR3-target count in *n*, only the first *n* CR3-target values are considered; if the CR3-target count is 0, MOV to CR3 always causes a VM exit.

  The first processors to support the virtual-machine extensions supported only the 1-setting of the "CR3-load exiting" VM-execution control. These processors always consult the CR3-target controls to determine whether an execution of MOV to CR3 causes a VM exit.
- **MOV to CR4.** The MOV to CR4 instruction causes a VM exit unless the value of its source operand matches, for the position of each bit set in the CR4 guest/host mask, the corresponding bit in the CR4 read shadow.
- **MOV to CR8.** The MOV to CR8 instruction (which can be executed only in 64-bit mode) causes a VM exit if the "CR8-load exiting" VM-execution control is 1. If this control is 0, the behavior of the MOV to CR8 instruction is modified if the "use TPR shadow" VM-execution control is 1 (see Section 21.4) and it may cause a trap-like VM exit (see below).
- **MOV DR.** The MOV DR instruction causes a VM exit if the "MOV-DR exiting" VM-execution control is 1. Such VM exits represent an exception to the principles identified in Section 21.1.1 in that they take priority over the following: general-protection exceptions based on privilege level; and invalid-opcode exceptions that occur because CR4.DE=1 and the instruction specified access to DR4 or DR5.
- **MWAIT.** The MWAIT instruction causes a VM exit if the "MWAIT exiting" VM-execution control is 1.
- **PAUSE.** The PAUSE instruction causes a VM exit if the "PAUSE exiting" VM-execution control is 1.
- **RDMSR.** The RDMSR instruction causes a VM exit if any of the following are true:
  - The "use MSR bitmaps" VM-execution control is 0.
  - The value of ECX is not in the range 00000000H – 00001FFFH or C0000000H – C0001FFFH.
  - The value of ECX is in the range 00000000H – 00001FFFH and bit *n* in read bitmap for low MSRs is 1, where *n* is the value of ECX.
  - The value of ECX is in the range C0000000H – C0001FFFH and bit *n* in read bitmap for high MSRs is 1, where *n* is the value of ECX & 00001FFFH.

  See Section 20.6.9 for details regarding how these bitmaps are identified.

- **RDPMC.** The RDPMC instruction causes a VM exit if the "RDPMC exiting" VM-execution control is 1.
- **RDTSC.** The RDTSC instruction causes a VM exit if the "RDTSC exiting" VM-execution control is 1.
- **RDTSCP.** The RDTSCP instruction causes a VM exit if the "RDTSC exiting" and "enable RDTSCP" VM-execution controls are both 1.
- **RSM.** The RSM instruction causes a VM exit if executed in system-management mode (SMM).[1]
- **WBINVD.** The WBINVD instruction causes a VM exit if the "WBINVD exiting" VM-execution control is 1.[2]
- **WRMSR.** The WRMSR instruction causes a VM exit if any of the following are true:
  - The "use MSR bitmaps" VM-execution control is 0.
  - The value of ECX is not in the range 00000000H – 00001FFFH or C0000000H – C0001FFFH.
  - The value of ECX is in the range 00000000H – 00001FFFH and bit *n* in write bitmap for low MSRs is 1, where *n* is the value of ECX.
  - The value of ECX is in the range C0000000H – C0001FFFH and bit *n* in write bitmap for high MSRs is 1, where *n* is the value of ECX & 00001FFFH.

  See Section 20.6.9 for details regarding how these bitmaps are identified.

  If an execution of WRMSR does not cause a VM exit as specified above and ECX = 808H (indicating the TPR MSR), instruction behavior is modified if the "virtualize x2APIC mode" VM-execution control is 1 (see Section 21.4) and it may cause a trap-like VM exit (see below).[3]

The MOV to CR8 and WRMSR instructions may cause "trap-like" VM exits. In such a case, the instruction completes before the VM exit occurs and that processor state is updated by the instruction (for example, the value of CS:RIP saved in the guest-state area of the VMCS references the next instruction).

1. Execution of the RSM instruction outside SMM causes an invalid-opcode exception regardless of whether the processor is in VMX operation. It also does so in VMX root operation in SMM; see Section 25.15.3.
2. "WBINVD exiting" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "WBINVD exiting" VM-execution control were 0. See Section 20.6.2.
3. "Virtualize x2APIC mode" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "virtualize x2APIC mode" VM-execution control were 0. See Section 20.6.2.

21.2 APIC-ACCESS VM EXITS

If the "virtualize APIC accesses" VM-execution control is 1, an attempt to access memory using a physical address on the APIC-access page (see Section 20.6.8) causes a VM exit.[1,2] Such a VM exit is called an **APIC-access VM exit**.

Whether an operation that attempts to access memory with a physical address on the APIC-access page causes an APIC-access VM exit may be qualified based on the type of access. Section 21.2.1 describes the treatment of linear accesses, while Section 21.2.3 describes that of physical accesses. Section 21.2.4 discusses accesses to the TPR field on the APIC-access page (called VTPR accesses), which do not, if the "use TPR shadow" VM-execution control is 1, cause APIC-access VM exits.

1. "Virtualize APIC accesses" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "virtualize APIC accesses" VM-execution control were 0. See Section 20.6.2.
2. Even when addresses are translated using EPT (see Chapter 24), the determination of whether an APIC-access VM exit occurs depends on an access's physical address, not its guest-physical address.

21.2.1.1 Linear Accesses That Cause APIC-Access VM Exits

Whether a linear access to the APIC-access page causes an APIC-access VM exit depends in part of the nature of the translation used by the linear address:

- If the linear access uses a translation with a 4-KByte page, it causes an APIC-access VM exit.
- If the linear access uses a translation with a large page (2-MByte or 4-MByte), the access may or may not cause an APIC-access VM exit. Section 21.5.1 describes the treatment of such accesses that do not cause an APIC-access VM exits.

  If CR0.PG = 1 and EPT is in use (the "enable EPT" VM-execution control is 1), a linear access uses a translation with a large page only if a large page is specified by both the guest paging structures and the EPT paging structures.[1]

It is recommended that software configure the paging structures so that any translation to the APIC-access page uses a 4-KByte page.

A linear access to the APIC-access page might not cause an APIC-access VM exit if the "enable EPT" VM-execution control is 1 and software has not properly invalidate information cached from the EPT paging structures:

- At time t_1, EPT was in use, the EPTP value was X, and some guest-physical address Y translated to an address that was not on the APIC-access page at that time. (This might be because the "virtualize APIC accesses" VM-execution control was 0.)
- At later time t_2, EPT is in use, the EPTP value is X, and a memory access uses a linear address that translates to Y, which now translates to an address on the APIC-access page. (This implies that the "virtualize APIC accesses" VM-execution control is 1 at this time.)
- Software did not execute the INVEPT instruction between times t_1 and t_2, either with the all-context INVEPT type or with the single-context INVEPT type and X as the INVEPT descriptor.

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, CR0.PG must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.

21.2.2 Guest-Physical Accesses to the APIC-Access Page

An access to the APIC-access page is called a **guest-physical access** if (1) CR0.PG = 1;[1] (2) the "enable EPT" VM-execution control is 1;[2] (3) the access's physical address is the result of an EPT translation; and (4) either (a) the access was not generated by a linear address; or (b) the access's guest-physical address is not the translation of the access's linear address. Guest-physical accesses include the following when guest-physical addresses are being translated using EPT:

- Reads from the guest paging structures when translating a linear address (such an access uses a guest-physical address that is not the translation of that linear address).
- Loads of the page-directory-pointer-table entries by MOV to CR when the logical processor is using (or that causes the logical processor to use) PAE paging.[3]
- Updates to the accessed and dirty bits in the guest paging structures when using a linear address (such an access uses a guest-physical address that is not the translation of that linear address).

Section 21.2.2.1 specifies when guest-physical accesses to the APIC-access page might not cause APIC-access VM exits. In general, the treatment of APIC-access VM exits caused by guest-physical accesses is similar to that of EPT violations. Based upon this treatment, Section 21.2.2.2 specifies the priority of such VM exits with respect to other events.

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, CR0.PG must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.
2. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.
3. A logical processor uses PAE paging if CR0.PG = 1, CR4.PAE = 1 and IA32_EFER.LMA = 0. See Section 3.8 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*.

21.2.3 Physical Accesses to the APIC-Access Page

An access to the APIC-access page is called a **physical access** if (1) either (a) the "enable EPT" VM-execution control is 0;[1] or (b) the access's physical address is not the result of a translation through the EPT paging structures; and (2) either (a) the access is not generated by a linear address; or (b) the access's physical address is not the translation of its linear address.

Physical accesses include the following:

- If the "enable EPT" VM-execution control is 0:
  - Reads from the paging structures when translating a linear address.
  - Loads of the page-directory-pointer-table entries by MOV to CR when the logical processor is using (or that causes the logical processor to use) PAE paging.[2]
  - Updates to the accessed and dirty bits in the paging structures.
- If the "enable EPT" VM-execution control is 1, accesses to the EPT paging structures.
- Any of the following accesses made by the processor to support VMX non-root operation:
  - Accesses to the VMCS region.
  - Accesses to data structures referenced (directly or indirectly) by physical addresses in VM-execution control fields in the VMCS. These include the I/O bitmaps, the MSR bitmaps, and the virtual-APIC page.
- Accesses that effect transitions into and out of SMM.[3] These include the following:
  - Accesses to SMRAM during SMI delivery and during execution of RSM.
  - Accesses during SMM VM exits (including accesses to MSEG) and during VM entries that return from SMM.

A physical access to the APIC-access page may or may not cause an APIC-access VM exit. (A physical write to the APIC-access page may write to memory as specified in Section 21.5.2 before causing the APIC-access VM exit.) The priority of an APIC-access VM exit caused by physical access is not defined relative to other events that the access may cause. Section 21.5.2 describes the treatment of physical accesses to the APIC-access page that do not cause APIC-access VM exits.

1. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.
2. A logical processor uses PAE paging if CR0.PG = 1, CR4.PAE = 1 and IA32_EFER.LMA = 0. See Section 3.8 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*.
3. Technically, these accesses do not occur in VMX non-root operation. They are included here for clarity.

21.4 CHANGES TO INSTRUCTION BEHAVIOR IN VMX NON-ROOT OPERATION

The behavior of some instructions is changed in VMX non-root operation. Some of these changes are determined by the settings of certain VM-execution control fields. The following items detail such changes:

- **CLTS.** Behavior of the CLTS instruction is determined by the bits in position 3 (corresponding to CR0.TS) in the CR0 guest/host mask and the CR0 read shadow:
  - If bit 3 in the CR0 guest/host mask is 0, CLTS clears CR0.TS normally (the value of bit 3 in the CR0 read shadow is irrelevant in this case), unless CR0.TS is fixed to 1 in VMX operation (see Section 19.8), in which case CLTS causes a general-protection exception.
  - If bit 3 in the CR0 guest/host mask is 1 and bit 3 in the CR0 read shadow is 0, CLTS completes but does not change the contents of CR0.TS.
  - If the bits in position 3 in the CR0 guest/host mask and the CR0 read shadow are both 1, CLTS causes a VM exit (see Section 21.1.3).
- **IRET.** Behavior of IRET with regard to NMI blocking (see Table 20-3) is determined by the settings of the "NMI exiting" and "virtual NMIs" VM-execution controls:
  - If the "NMI exiting" VM-execution control is 0, IRET operates normally and unblocks NMIs.
  - If the "NMI exiting" VM-execution control is 1, IRET does not affect blocking of NMIs.
  - If the "virtual NMIs" VM-execution control is 1, the logical processor tracks virtual-NMI blocking. In this case, IRET removes any virtual-NMI blocking.

  If the "NMI exiting" VM-execution control is 0, the "virtual NMIs" control must be 0. (See Section 22.2.1.1.)
- **LMSW.** Outside of VMX non-root operation, LMSW loads its source operand into CR0[3:0], but it does not clear CR0.PE if that bit is set. In VMX non-root operation, an execution of LMSW that does not cause a VM exit (see Section 21.1.3) leaves unmodified any bit in CR0[3:0] corresponding to a bit set in the CR0 guest/host mask. An attempt to set any other bit in CR0[3:0] to a value not supported in VMX operation (see Section 19.8) causes a general-protection exception. Attempts to clear CR0.PE are ignored without fault.
- **MOV from CR0.** The behavior of MOV from CR0 is determined by the CR0 guest/host mask and the CR0 read shadow. For each position corresponding to a bit clear in the CR0 guest/host mask, the destination operand is loaded with the value of the corresponding bit in CR0. For each position corresponding to a bit set in the CR0 guest/host mask, the destination operand is loaded with the value of the corresponding bit in the CR0 read shadow. Thus, if every bit is cleared in the CR0 guest/host mask, MOV from CR0 reads normally from CR0; if every bit is set in the CR0 guest/host mask, MOV from CR0 returns the value of the CR0 read shadow.

  Depending on the contents of the CR0 guest/host mask and the CR0 read shadow, bits may be set in the destination that would never be set when reading directly from CR0.
- **MOV from CR3.** If the "enable EPT" VM-execution control is 1 and an execution of MOV from CR3 does not cause a VM exit (see Section 21.1.3), the value loaded from CR3 is a guest-physical address; see Section 24.2.1.

- **MOV from CR4.** The behavior of MOV from CR4 is determined by the CR4 guest/host mask and the CR4 read shadow. For each position corresponding to a bit clear in the CR4 guest/host mask, the destination operand is loaded with the value of the corresponding bit in CR4. For each position corresponding to a bit set in the CR4 guest/host mask, the destination operand is loaded with the value of the corresponding bit in the CR4 read shadow. Thus, if every bit is cleared in the CR4 guest/host mask, MOV from CR4 reads normally from CR4; if every bit is set in the CR4 guest/host mask, MOV from CR4 returns the value of the CR4 read shadow.

  Depending on the contents of the CR4 guest/host mask and the CR4 read shadow, bits may be set in the destination that would never be set when reading directly from CR4.
- **MOV from CR8.** Behavior of the MOV from CR8 instruction (which can be executed only in 64-bit mode) is determined by the settings of the "CR8-store exiting" and "use TPR shadow" VM-execution controls:
  - If both controls are 0, MOV from CR8 operates normally.
  - If the "CR8-store exiting" VM-execution control is 0 and the "use TPR shadow" VM-execution control is 1, MOV from CR8 reads from the TPR shadow. Specifically, it loads bits 3:0 of its destination operand with the value of bits 7:4 of byte 80H of the virtual-APIC page (see Section 20.6.8). Bits 63:4 of the destination operand are cleared.
  - If the "CR8-store exiting" VM-execution control is 1, MOV from CR8 causes a VM exit (see Section 21.1.3); the "use TPR shadow" VM-execution control is ignored in this case.
- **MOV to CR0.** An execution of MOV to CR0 that does not cause a VM exit (see Section 21.1.3) leaves unmodified any bit in CR0 corresponding to a bit set in the CR0 guest/host mask. Treatment of attempts to modify other bits in CR0 depends on the setting of the "unrestricted guest" VM-execution control:[1]
  - If the control is 0, MOV to CR0 causes a general-protection exception if it attempts to set any bit in CR0 to a value not supported in VMX operation (see Section 19.8).
  - If the control is 1, MOV to CR0 causes a general-protection exception if it attempts to set any bit in CR0 other than bit 0 (PE) or bit 31 (PG) to a value not supported in VMX operation. It remains the case, however, that MOV to CR0 causes a general-protection exception if it would result in CR0.PE = 0 and CR0.PG = 1 or if it would result in CR0.PG = 1, CR4.PAE = 0, and IA32_EFER.LME = 1.
- **MOV to CR3.** If the "enable EPT" VM-execution control is 1 and an execution of MOV to CR3 does not cause a VM exit (see Section 21.1.3), the value loaded into CR3 is treated as a guest-physical address; see Section 24.2.1.
  - If PAE paging is not being used, the instruction does not use the guest-physical address to access memory and it does not cause it to be translated through EPT.[2]
  - If PAE paging is being used, the instruction translates the guest-physical address through EPT and uses the result to load the four (4) page-directory-pointer-table

1. "Unrestricted guest" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "unrestricted guest" VM-execution control were 0. See Section 20.6.2.
2. A logical processor uses PAE paging if CR0.PG = 1, CR4.PAE = 1 and IA32_EFER.LMA = 0. See Section 3.8 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*.

entries (PDPTEs). The instruction does not use the guest-physical addresses the PDPTEs to access memory and it does not cause them to be translated through EPT.

- **MOV to CR4.** An execution of MOV to CR4 that does not cause a VM exit (see Section 21.1.3) leaves unmodified any bit in CR4 corresponding to a bit set in the CR4 guest/host mask. Such an execution causes a general-protection exception if it attempts to set any bit in CR4 (not corresponding to a bit set in the CR4 guest/host mask) to a value not supported in VMX operation (see Section 19.8).
- **MOV to CR8.** Behavior of the MOV to CR8 instruction (which can be executed only in 64-bit mode) is determined by the settings of the "CR8-load exiting" and "use TPR shadow" VM-execution controls:
  - If both controls are 0, MOV to CR8 operates normally.
  - If the "CR8-load exiting" VM-execution control is 0 and the "use TPR shadow" VM-execution control is 1, MOV to CR8 writes to the TPR shadow. Specifically, it stores bits 3:0 of its source operand into bits 7:4 of byte 80H of the virtual-APIC page (see Section 20.6.8); bits 3:0 of that byte and bytes 129-131 of that page are cleared. Such a store may cause a VM exit to occur after it completes (see Section 21.1.3).
  - If the "CR8-load exiting" VM-execution control is 1, MOV to CR8 causes a VM exit (see Section 21.1.3); the "use TPR shadow" VM-execution control is ignored in this case.
- **RDMSR.** Section 21.1.3 identifies when executions of the RDMSR instruction cause VM exits. If such an execution causes neither a fault due to CPL > 0 nor a VM exit, the instruction's behavior may be modified for certain values of ECX:
  - If ECX contains 10H (indicating the IA32_TIME_STAMP_COUNTER MSR), the value returned by the instruction is determined by the setting of the "use TSC offsetting" VM-execution control as well as the TSC offset:
    - If the control is 0, the instruction operates normally, loading EAX:EDX with the value of the IA32_TIME_STAMP_COUNTER MSR.
    - If the control is 1, the instruction loads EAX:EDX with the sum (using signed addition) of the value of the IA32_TIME_STAMP_COUNTER MSR and the value of the TSC offset (interpreted as a signed value).
  - If ECX contains 808H (indicating the TPR MSR), instruction behavior is determined by the setting of the "virtualize x2APIC mode" VM-execution control:[1]
    - If the control is 0, the instruction operates normally. If the local APIC is in x2APIC mode, EAX[7:0] is loaded with the value of the APIC's task-priority register (EDX and EAX[31:8] are cleared to 0). If the local APIC is not in x2APIC mode, a general-protection fault occurs.
    - If the control is 1, the instruction loads EAX:EDX with the value of bytes 87H:80H of the virtual-APIC page. This occurs even if the local APIC is not in x2APIC mode (no general-protection fault occurs because the local APIC is not x2APIC mode).

1. "Virtualize x2APIC mode" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "virtualize x2APIC mode" VM-execution control were 0. See Section 20.6.2.

- **RDTSC.** Behavior of the RDTSC instruction is determined by the settings of the "RDTSC exiting" and "use TSC offsetting" VM-execution controls as well as the TSC offset:
  - If both controls are 0, RDTSC operates normally.
  - If the "RDTSC exiting" VM-execution control is 0 and the "use TSC offsetting" VM-execution control is 1, RDTSC loads EAX:EDX with the sum (using signed addition) of the value of the IA32_TIME_STAMP_COUNTER MSR and the value of the TSC offset (interpreted as a signed value).
  - If the "RDTSC exiting" VM-execution control is 1, RDTSC causes a VM exit (see Section 21.1.3).
- **RDTSCP.** Behavior of the RDTSCP instruction is determined first by the setting of the "enable RDTSCP" VM-execution control:[1]
  - If the "enable RDTSCP" VM-execution control is 0, RDTSCP causes an invalid-opcode exception (#UD).
  - If the "enable RDTSCP" VM-execution control is 1, treatment is based on the settings the "RDTSC exiting" and "use TSC offsetting" VM-execution controls as well as the TSC offset:
    - If both controls are 0, RDTSCP operates normally.
    - If the "RDTSC exiting" VM-execution control is 0 and the "use TSC offsetting" VM-execution control is 1, RDTSCP loads EAX:EDX with the sum (using signed addition) of the value of the IA32_TIME_STAMP_COUNTER MSR and the value of the TSC offset (interpreted as a signed value); it also loads ECX with the value of bits 31:0 of the IA32_TSC_AUX MSR.

1. "Enable RDTSCP" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "enable RDTSCP" VM-execution control were 0. See Section 20.6.2.

21.5 APIC ACCESSES THAT DO NOT CAUSE VM EXITS

As noted in Section 21.2, if the "virtualize APIC accesses" VM-execution control is 1, most memory accesses to the APIC-access page (see Section 20.6.8) cause APIC-access VM exits.[1] Section 21.2 identifies potential exceptions. These are covered in Section 21.5.1 through Section 21.5.3.

In some cases, an attempt to access memory on the APIC-access page is converted to an access to the virtual-APIC page (see Section 20.6.8). In these cases, the access uses the memory type reported in bit 53:50 of the IA32_VMX_BASIC MSR (see Appendix G.1).

1. "Virtualize APIC accesses" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "virtualize APIC accesses" VM-execution control were 0. See Section 20.6.2.

21.8 UNRESTRICTED GUESTS

The first processors to support VMX operation require CR0.PE and CR0.PG to be 1 in VMX operation (see Section 19.8). This restriction implies that guest software cannot be run in unpaged protected mode or in real-address mode. Later processors support a VM-execution control called "unrestricted guest".[1] If this control is 1, CR0.PE and CR0.PG may be 0 in VMX non-root operation. Such processors allow guest software to run in unpaged protected mode or in real-address mode. The following items describe the behavior of such software:

- The MOV CR0 instructions does not cause a general-protection exception simply because it would set either CR0.PE and CR0.PG to 0. See Section 21.4 for details.
- A logical processor treats the values of CR0.PE and CR0.PG in VMX non-root operation just as it does outside VMX operation. Thus, if CR0.PE = 0, the processor operates as it does normally in real-address mode (for example, it uses the 16-bit **interrupt table** to deliver interrupts and exceptions). If CR0.PG = 0, the processor operates as it does normally when paging is disabled.
- Processor operation is modified by the fact that the processor is in VMX non-root operation and by the settings of the VM-execution controls just as it is in protected mode or when paging is enabled. Instructions, interrupts, and exceptions that cause VM exits in protected mode or when paging is enabled also do so in real-address mode or when paging is disabled. The following examples should be noted:
  - If CR0.PG = 0, page faults do not occur and thus cannot cause VM exits.
  - If CR0.PE = 0, invalid-TSS exceptions do not occur and thus cannot cause VM exits.
  - If CR0.PE = 0, the following instructions cause invalid-opcode exceptions and do not cause VM exits: INVEPT, INVVPID, LLDT, LTR, SLDT, STR, VMCLEAR, VMLAUNCH, VMPTRLD, VMPTRST, VMREAD, VMRESUME, VMWRITE, VMXOFF, and VMXON.
- If CR0.PG = 0, each linear address is passed directly to the EPT mechanism for translation to a physical address.[2] The guest memory type passed on to the EPT mechanism is WB (writeback).

1. "Unrestricted guest" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "unrestricted guest" VM-execution control were 0. See Section 20.6.2.
2. As noted in Section 22.2.1.1, the "enable EPT" VM-execution control must be 1 if the "unrestricted guest" VM-execution control is 1.

14. Real-mode Virtualization Updates to Chapter 22 (Chapter 22, Vol 3B)

Change bars show changes to Chapter 22 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

22.2.1.1 VM-Execution Control Fields

VM entries perform the following checks on the VM-execution control fields:[1]

- Reserved bits in the pin-based VM-execution controls must be set properly. Software may consult the VMX capability MSRs to determine the proper settings (see Appendix G.3.1).
- Reserved bits in the primary processor-based VM-execution controls must be set properly. Software may consult the VMX capability MSRs to determine the proper settings (see Appendix G.3.2).
- If the "activate secondary controls" primary processor-based VM-execution control is 1, reserved bits in the secondary processor-based VM-execution controls must be set properly. Software may consult the VMX capability MSRs to determine the proper settings (see Appendix G.3.3).

  If the "activate secondary controls" primary processor-based VM-execution control is 0 (or if the processor does not support the 1-setting of that control), no checks are performed on the secondary processor-based VM-execution controls. The logical processor operates as if all the secondary processor-based VM-execution controls were 0.
- The CR3-target count must not be greater than 4. Future processors may support a different number of CR3-target values. Software should read the VMX capability MSR IA32_VMX_MISC to determine the number of values supported (see Appendix G.6).

1. If the "activate secondary controls" primary processor-based VM-execution control is 0, VM entry operates as if each secondary processor-based VM-execution control were 0.

- If the "use I/O bitmaps" VM-execution control is 1, bits 11:0 of each I/O-bitmap address must be 0. On processors that support Intel 64 architecture, neither address should set any bits beyond the processor's physical-address width.[1] On processors that do not support Intel 64 architecture, neither address should set any bits in the range 63:32.
- If the "use MSR bitmaps" VM-execution control is 1, bits 11:0 of the MSR-bitmap address must be 0. On processors that support Intel 64 architecture, the address should not set any bits beyond the processor's physical-address width. On processors that do not support Intel 64 architecture, the address should not set any bits in the range 63:32.
- If the "use TPR shadow" VM-execution control is 1, the virtual-APIC address must satisfy the following checks:
  - Bits 11:0 of the address must be 0.
  - On processors that support Intel 64 architecture, the address should not set any bits beyond the processor's physical-address width.
  - On processors that support the IA-32 architecture, the address should not set any bits in the range 63:32.

  The following items describe the treatment of bytes 81H-83H on the virtual-APIC page (see Section 20.6.8) if all of the above checks are satisfied and the "use TPR shadow" VM-execution control is 1, treatment depends upon the setting of the "virtualize APIC accesses" VM-execution control:[2]
  - If the "virtualize APIC accesses" VM-execution control is 0, the bytes may be cleared. (If the bytes are not cleared, they are left unmodified.)
  - If the "virtualize APIC accesses" VM-execution control is 1, the bytes are cleared.
  - Any clearing of the bytes occurs even if the VM entry subsequently fails.
- If the "use TPR shadow" VM-execution control is 1, bits 31:4 of the TPR threshold VM-execution control field must be 0.
- The following check is performed if the "use TPR shadow" VM-execution control is 1 and the "virtualize APIC accesses" VM-execution control is 0: the value of bits 3:0 of the TPR threshold VM-execution control field should not be greater than the value of bits 7:4 in byte 80H on the virtual-APIC page (see Section 20.6.8).
- If the "NMI exiting" VM-execution control is 0, the "virtual NMIs" VM-execution control must be 0.
- If the "virtual NMIs" VM-execution control is 0, the "NMI-window exiting" VM-execution control must be 0.
- If the "virtualize APIC-accesses" VM-execution control is 1, the APIC-access address must satisfy the following checks:
  - Bits 11:0 of the address must be 0.
  - On processors that support Intel 64 architecture, the address should not set any bits beyond the processor's physical-address width.

---

1. Software can determine a processor's physical-address width by executing CPUID with 80000008H in EAX. The physical-address width is returned in bits 7:0 of EAX.
2. "Virtualize APIC accesses" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "virtualize APIC accesses" VM-execution control were 0. See Section 20.6.2.

  - On processors that support the IA-32 architecture, the address should not set any bits in the range 63:32.
- If the "virtualize x2APIC mode" VM-execution control is 1, the "use TPR shadow" VM-execution control must be 1 and the "virtualize APIC accesses" VM-execution control must be 0.[1]
- If the "enable VPID" VM-execution control is 1, the value of the VPID VM-execution control field must not be 0000H.
- If the "enable EPT" VM-execution control is 1, the EPTP VM-execution control field (see Table 20-8 in Section 20.6.11) must satisfy the following checks:[2]
  - The EPT memory type (bits 2:0) must be a value supported by the logical processor as indicated in the IA32_VMX_EPT_VPID_CAP MSR (see Appendix G.10).
  - Bits 5:3 (1 less than the EPT page-walk length) must be 3, indicating an EPT page-walk length of 4; see Section 24.2.2.
  - Reserved bits 11:6 and 63:N (where N is the processor's physical-address width) must all be 0.
  - If the "unrestricted guest" VM-execution control is 1, the "enable EPT" VM-execution control must also be 1.[3]

1. "Virtualize APIC accesses" and "virtualize x2APIC mode" are both secondary processor-based VM-execution controls. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if both these controls were 0. See Section 20.6.2.
2. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.
3. "Unrestricted guest" and "enable EPT" are both secondary processor-based VM-execution controls. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if both these controls were 0. See Section 20.6.2.

22.2.1.3 VM-Entry Control Fields

VM entries perform the following checks on the VM-entry control fields.

- Reserved bits in the VM-entry controls must be set properly. Software may consult the VMX capability MSRs to determine the proper settings (see Appendix G.5).
- Fields relevant to VM-entry event injection must be set properly. These fields are the VM-entry interruption-information field (see Table 20-12 in Section 20.8.3), the VM-entry exception error code, and the VM-entry instruction length. If the valid bit (bit 31) in the VM-entry interruption-information field is 1, the following must hold:
  - The field's interruption type (bits 10:8) is not set to a reserved value. Value 1 is reserved on all logical processors; value 7 (other event) is reserved on logical processors that do not support the 1-setting of the "monitor trap flag" VM-execution control.
  - The field's vector (bits 7:0) is consistent with the interruption type:
    - If the interruption type is non-maskable interrupt (NMI), the vector is 2.
    - If the interruption type is hardware exception, the vector is at most 31.
    - If the interruption type is other event, the vector is 0 (pending MTF VM exit).
  - The field's deliver-error-code bit (bit 11) is 1 if and only if (1) bit 0 (corresponding to CR0.PE) is set in the CR0 field in the guest-state area; (2) the interruption type is hardware exception; and (3) the vector indicates an exception that would normally deliver an error code (8 = #DF; 10 = TS; 11 = #NP; 12 = #SS; 13 = #GP; 14 = #PF; or 17 = #AC).[1]
  - Reserved bits in the field (30:12) are 0.
  - If the deliver-error-code bit (bit 11) is 1, bits 31:15 of the VM-entry exception error-code field are 0.
  - If the interruption type is software interrupt, software exception, or privileged software exception, the VM-entry instruction-length field is in the range 1–15.
- The following checks are performed for the VM-entry MSR-load address if the VM-entry MSR-load count field is non-zero:
  - The lower 4 bits of the VM-entry MSR-load address must be 0. On processors that support Intel 64 architecture, the address should not set any bits beyond the processor's physical-address width.[2] On processors that do not support Intel 64 architecture, the address should not set any bits in the range 63:32.

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation, bit 0 in the CR0 field in the guest-state area must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.
2. Software can determine a processor's physical-address width by executing CPUID with 80000008H in EAX. The physical-address width is returned in bits 7:0 of EAX.

22.2.2 Checks on Host Control Registers and MSRs

The following checks are performed on fields in the host-state area that correspond to control registers and MSRs:

- The CR0 field must not set any bit to a value not supported in VMX operation (see Section 19.8).[1]

1. The bits corresponding to CR0.NW (bit 29) and CR0.CD (bit 30) are never checked because the values of these bits are not changed by VM exit; see Section 23.5.1.

22.3.1.1 Checks on Guest Control Registers, Debug Registers, and MSRs

The following checks are performed on fields in the guest-state area corresponding to control registers, debug registers, and MSRs:

- The CR0 field must not set any bit to a value not supported in VMX operation (see Section 19.8). The following are exceptions:
  - Bit 0 (corresponding to CR0.PE) and bit 31 (PG) are not checked if the "unrestricted guest" VM-execution control is 1.[1]
  - Bit 29 (corresponding to CR0.NW) and bit 30 (CD) are never checked because the values of these bits are not changed by VM entry; see Section 22.3.2.1.
- If bit 31 in the CR0 field (corresponding to PG) is 1, bit 0 in that field (PE) must also be 1.[2]
- The CR4 field must not set any bit to a value not supported in VMX operation (see Section 19.8).
- If the "load debug controls" VM-entry control is 1, bits reserved in the IA32_DEBUGCTL MSR must be 0 in the field for that register. The first processors to support the virtual-machine extensions supported only the 1-setting of this control and thus performed this check unconditionally.
- The following checks are performed on processors that support Intel 64 architecture:
  - If the "IA-32e mode guest" VM-entry control is 1, bit 31 in the CR0 field (corresponding to CR0.PG) and bit 5 in the CR4 field (corresponding to CR4.PAE) must each be 1.[3]

1. "Unrestricted guest" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "unrestricted guest" VM-execution control were 0. See Section 20.6.2.
2. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation, bit 0 in the CR0 field must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.
3. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, bit 31 in the CR0 field must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

  - The CR3 field must be such that bits 63:52 and bits in the range 51:32 beyond the processor's physical-address width are 0.[1]
  - If the "load debug controls" VM-entry control is 1, bits 63:32 in the DR7 field must be 0. The first processors to support the virtual-machine extensions supported only the 1-setting of this control and thus performed this check unconditionally (if they supported Intel 64 architecture).
  - The IA32_SYSENTER_ESP field and the IA32_SYSENTER_EIP field must each contain a canonical address.
- If the "load IA32_PERF_GLOBAL_CTRL" VM-entry control is 1, bits reserved in the IA32_PERF_GLOBAL_CTRL MSR must be 0 in the field for that register (see Figure 18-15).
- If the "load IA32_PAT" VM-entry control is 1, the value of the field for the IA32_PAT MSR must be one that could be written by WRMSR without fault at CPL 0. Specifically, each of the 8 bytes in the field must have one of the values 0 (UC), 1 (WC), 4 (WT), 5 (WP), 6 (WB), or 7 (UC-).
- If the "load IA32_EFER" VM-entry control is 1, the following checks are performed on the field for the IA32_EFER MSR :
  - Bits reserved in the IA32_EFER MSR must be 0.
  - Bit 10 (corresponding to IA32_EFER.LMA) must equal the value of the "IA-32e mode guest" VM-exit control. It must also be identical to bit 8 (LME) if bit 31 in the CR0 field (corresponding to CR0.PG) is 1.[2]

1. Software can determine a processor's physical-address width by executing CPUID with 80000008H in EAX. The physical-address width is returned in bits 7:0 of EAX.
2. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, bit 31 in the CR0 field must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

22.3.1.2 Checks on Guest Segment Registers

This section specifies the checks on the fields for CS, SS, DS, ES, FS, GS, TR, and LDTR. The following terms are used in defining these checks:

- The guest will be **virtual-8086** if the VM flag (bit 17) is 1 in the RFLAGS field in the guest-state area.
- The guest will be **IA-32e mode** if the "IA-32e mode guest" VM-entry control is 1. (This is possible only on processors that support Intel 64 architecture.)
- Any one of these registers is said to be **usable** if the unusable bit (bit 16) is 0 in the access-rights field for that register.

The following are the checks on these fields:

- Selector fields.
  - TR. The TI flag (bit 2) must be 0.
  - LDTR. If LDTR is usable, the TI flag (bit 2) must be 0.
  - SS. If the guest will not be virtual-8086 and the "unrestricted guest" VM-execution control is 0, the RPL (bits 1:0) must equal the RPL of the selector field for CS.[1]
- Base-address fields.
  - CS, SS, DS, ES, FS, GS. If the guest will be virtual-8086, the address must be the selector field shifted left 4 bits (multiplied by 16).
  - The following checks are performed on processors that support Intel 64 architecture:
    - TR, FS, GS. The address must be canonical.
    - LDTR. If LDTR is usable, the address must be canonical.
    - CS. Bits 63:32 of the address must be zero.
    - SS, DS, ES. If the register is usable, bits 63:32 of the address must be zero.
- Limit fields for CS, SS, DS, ES, FS, GS. If the guest will be virtual-8086, the field must be 0000FFFFH.
- Access-rights fields.
  - CS, SS, DS, ES, FS, GS.
    - If the guest will be virtual-8086, the field must be 000000F3H. This implies the following:
      - Bits 3:0 (Type) must be 3, indicating an expand-up read/write accessed data segment.
      - Bit 4 (S) must be 1.
      - Bits 6:5 (DPL) must be 3.
      - Bit 7 (P) must be 1.
      - Bits 11:8 (reserved), bit 12 (software available), bit 13 (reserved/L), bit 14 (D/B), bit 15 (G), bit 16 (unusable), and bits 31:17 (reserved) must all be 0.

1. "Unrestricted guest" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "unrestricted guest" VM-execution control were 0. See Section 20.6.2.

- If the guest will not be virtual-8086, the different sub-fields are considered separately:
  - Bits 3:0 (Type).
    - CS. The values allowed depend on the setting of the "unrestricted guest" VM-execution control:
      - If the control is 0, the Type must be 9, 11, 13, or 15 (accessed code segment).
      - If the control is 1, the Type must be either 3 (read/write accessed expand-up data segment) or one of 9, 11, 13, and 15 (accessed code segment).
    - SS. If SS is usable, the Type must be 3 or 7 (read/write, accessed data segment).
    - DS, ES, FS, GS. The following checks apply if the register is usable:
      - Bit 0 of the Type must be 1 (accessed).
      - If bit 3 of the Type is 1 (code segment), then bit 1 of the Type must be 1 (readable).
  - Bit 4 (S). If the register is CS or if the register is usable, S must be 1.
  - Bits 6:5 (DPL).
    - CS.
      - If the Type is 3 (read/write accessed expand-up data segment), the DPL must be 0. The Type can be 3 only if the "unrestricted guest" VM-execution control is 1.
      - If the Type is 9 or 11 (non-conforming code segment), the DPL must equal the DPL in the access-rights field for SS.
      - If the Type is 13 or 15 (conforming code segment), the DPL cannot be greater than the DPL in the access-rights field for SS.
    - SS.
      - If the "unrestricted guest" VM-execution control is 0, the DPL must equal the RPL from the selector field.
      - The DPL must be 0 either if the Type in the access-rights field for CS is 3 (read/write accessed expand-up data segment) or if bit 0 in the CR0 field (corresponding to CR0.PE) is 0.[1]
    - DS, ES, FS, GS. The DPL cannot be less than the RPL in the selector field if (1) the "unrestricted guest" VM-execution control is 0; (2) the register is usable; and (3) the Type in the access-rights field is in the range 0 – 11 (data segment or non-conforming code segment).
  - Bit 7 (P). If the register is CS or if the register is usable, P must be 1.
  - Bits 11:8 (reserved). If the register is CS or if the register is usable, these bits must all be 0.

1. The following apply if either the "unrestricted guest" VM-execution control or bit 31 of the primary processor-based VM-execution controls is 0: (1) bit 0 in the CR0 field must be 1 if the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation; and (2) the Type in the access-rights field for CS cannot be 3.

  - Bit 14 (D/B). For CS, D/B must be 0 if the guest will be IA-32e mode and the L bit (bit 13) in the access-rights field is 1.
  - Bit 15 (G). The following checks apply if the register is CS or if the register is usable:
    - If any bit in the limit field in the range 11:0 is 0, G must be 0.
    - If any bit in the limit field in the range 31:20 is 1, G must be 1.
  - Bits 31:17 (reserved). If the register is CS or if the register is usable, these bits must all be 0.

- TR. The different sub-fields are considered separately:
  - Bits 3:0 (Type).
    - If the guest will not be IA-32e mode, the Type must be 3 (16-bit busy TSS) or 11 (32-bit busy TSS).
    - If the guest will be IA-32e mode, the Type must be 11 (64-bit busy TSS).
  - Bit 4 (S). S must be 0.
  - Bit 7 (P). P must be 1.
  - Bits 11:8 (reserved). These bits must all be 0.
  - Bit 15 (G).
    - If any bit in the limit field in the range 11:0 is 0, G must be 0.
    - If any bit in the limit field in the range 31:20 is 1, G must be 1.
  - Bit 16 (Unusable). The unusable bit must be 0.
  - Bits 31:17 (reserved). These bits must all be 0.

22.3.1.4 Checks on Guest RIP and RFLAGS

The following checks are performed on fields in the guest-state area corresponding to RIP and RFLAGS:

- RIP. The following checks are performed on processors that support Intel 64 architecture:
  - Bits 63:32 must be 0 if the "IA-32e mode guest" VM-entry control is 0 or if the L bit (bit 13) in the access-rights field for CS is 0.
  - If the processor supports N < 64 linear-address bits, bits 63:N must be identical if the "IA-32e mode guest" VM-entry control is 1 and the L bit in the access-rights field for CS is 1.[1] (No check applies if the processor supports 64 linear-address bits.)
- RFLAGS.
  - Reserved bits 63:22 (bits 31:22 on processors that do not support Intel 64 architecture), bit 15, bit 5 and bit 3 must be 0 in the field, and reserved bit 1 must be 1.
  - The VM flag (bit 17) must be 0 either if the "IA-32e mode guest" VM-entry control is 1 or if bit 0 in the CR0 field (corresponding to CR0.PE) is 0.[2]

1. Software can determine the number N by executing CPUID with 80000008H in EAX. The number of linear-address bits supported is returned in bits 15:8 of EAX.
2. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation, bit 0 in the CR0 field must be 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

22.3.1.6 Checks on Guest Page-Directory-Pointer-Table Entries

If CR0.PG =1 and CR4.PAE = 1, the logical processor uses the **physical-address extension** (PAE). If IA32_EFER.LMA = 0, the logical processor also uses **PAE paging** (see Section 3.8 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*).[1] When PAE paging is in use, the physical address in CR3 references a table of **page-directory-pointer-table entries** (PDPTEs). A MOV to CR3 when PAE paging is in use checks the validity of the PDPTEs.

A VM entry is to a guest that uses PAE paging if (1) bit 5 (corresponding to CR4.PAE) is set in the CR4 field in the guest-state area; and (2) the "IA-32e mode guest" VM-entry control is 0. Such a VM entry checks the validity of the PDPTEs:

- If the "enable EPT" VM-execution control is 0, VM entry checks the validity of the PDPTEs referenced by the CR3 field in the guest-state area if either (1) PAE paging was not in use before the VM entry; or (2) the value of CR3 is changing as a result of the VM entry. VM entry may check their validity even if neither (1) nor (2) hold.[2]
- If the "enable EPT" VM-execution control is 1, VM entry checks the validity of the PDPTE fields in the guest-state area (see Section 20.4.2).

A VM entry to a guest that does not use PAE paging does not check the validity of any PDPTEs.

A VM entry that checks the validity of the PDPTEs uses the same checks that are used when CR3 is loaded with MOV to CR3 when PAE paging is in use.[3] If MOV to CR3 would cause a general-protection exception due to the PDPTEs that would be loaded (e.g., because a reserved bit is set), the VM entry fails.

1. On processors that support Intel 64 architecture, the physical-address extension may support more than 36 physical-address bits. Software can determine the number physical-address bits supported by executing CPUID with 80000008H in EAX. The physical-address width is returned in bits 7:0 of EAX.
2. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.
3. This implies that (1) bits 11:9 in each PDPTE are ignored; and (2) if bit 0 (present) is clear in one of the PDPTEs, bits 63:1 of that PDPTE are ignored.

22.3.2.1 Loading Guest Control Registers, Debug Registers, and MSRs

The following items describe how guest control registers, debug registers, and MSRs are loaded on VM entry:

- CR0 is loaded from the CR0 field with the exception of the following bits, which are never modified on VM entry: ET (bit 4); reserved bits 15:6, 17, and 28:19; NW (bit 29) and CD (bit 30).[1] The values of these bits in the CR0 field are ignored.
- CR3 and CR4 are loaded from the CR3 field and the CR4 field, respectively.
- If the "load debug controls" VM-execution control is 1, DR7 is loaded from the DR7 field with the exception that bit 12 and bits 15:14 are always 0 and bit 10 is always 1. The values of these bits in the DR7 field are ignored.

  The first processors to support the virtual-machine extensions supported only the 1-setting of the "load debug controls" VM-execution control and thus always loaded DR7 from the DR7 field.
- The following describes how some MSRs are loaded using fields in the guest-state area:
  - If the "load debug controls" VM-execution control is 1, the IA32_DEBUGCTL MSR is loaded from the IA32_DEBUGCTL field. The first processors to support the virtual-machine extensions supported only the 1-setting of this control and thus always loaded the IA32_DEBUGCTL MSR from the IA32_DEBUGCTL field.
  - The IA32_SYSENTER_CS MSR is loaded from the IA32_SYSENTER_CS field. Since this field has only 32 bits, bits 63:32 of the MSR are cleared to 0.
  - The IA32_SYSENTER_ESP and IA32_SYSENTER_EIP MSRs are loaded from the IA32_SYSENTER_ESP field and the IA32_SYSENTER_EIP field, respectively. On processors that do not support Intel 64 architecture, these fields have only 32 bits; bits 63:32 of the MSRs are cleared to 0.
  - The following are performed on processors that support Intel 64 architecture:
    - The MSRs FS.base and GS.base are loaded from the base-address fields for FS and GS, respectively (see Section 22.3.2.2).
    - If the "load IA32_EFER" VM-entry control is 0, bits in the IA32_EFER MSR are modified as follows:
      - IA32_EFER.LMA is loaded with the setting of the "IA-32e mode guest" VM-entry control.
      - If CR0 is being loaded so that CR0.PG = 1, IA32_EFER.LME is also loaded with the setting of the "IA-32e mode guest" VM-entry control.[2] Otherwise, IA32_EFER.LME is unmodified.

      See below for the case in which the "load IA32_EFER" VM-entry control is 1

1. Bits 15:6, bit 17, and bit 28:19 of CR0 and CR0.ET are unchanged by executions of MOV to CR0. Bits 15:6, bit 17, and bit 28:19 of CR0 are always 0 and CR0.ET is always 1.
2. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, VM entry must be loading CR0 so that CR0.PG = 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

22.3.2.2 Loading Guest Segment Registers and Descriptor-Table Registers

For each of CS, SS, DS, ES, FS, GS, TR, and LDTR, fields are loaded from the guest-state area as follows:

- The unusable bit is loaded from the access-rights field. This bit can never be set for TR (see Section 22.3.1.2). If it is set for one of the other registers, the following apply:
  - For each of CS, SS, DS, ES, FS, and GS, uses of the segment cause faults (general-protection exception or stack-fault exception) outside 64-bit mode, just as they would had the segment been loaded using a null selector. This bit does not cause accesses to fault in 64-bit mode.
  - If this bit is set for LDTR, uses of LDTR cause general-protection exceptions in all modes, just as they would had LDTR been loaded using a null selector.

  If this bit is clear for any of CS, SS, DS, ES, FS, GS, TR, and LDTR, a null selector value does not cause a fault (general-protection exception or stack-fault exception).
- TR. The selector, base, limit, and access-rights fields are loaded.
- CS.
  - The following fields are always loaded: selector, base address, limit, and (from the access-rights field) the L, D, and G bits.
  - For the other fields, the unusable bit of the access-rights field is consulted:
    - If the unusable bit is 0, all of the access-rights fields are loaded.
    - If the unusable bit is 1, the remainder of CS access rights are undefined after VM entry.
- SS, DS, ES, FS, and GS, and LDTR.
  - The selector fields are loaded.
  - For the other fields, the unusable bit of the corresponding access-rights field is consulted:
    - If the unusable bit is 0, the base-address, limit, and access-rights fields are loaded.
    - If the unusable bit is 1, the base address, the segment limit, and the remainder of the access rights are undefined after VM entry. The only exceptions are the following:
      - SS.DPL: always loaded from the SS access-rights field. This will be the current privilege level (CPL) after the VM entry completes.
      - The base addresses for FS and GS: always loaded. On processors that support Intel 64 architecture, the values loaded for base addresses for FS and GS are also manifest in the FS.base and GS.base MSRs.
      - The base address for LDTR on processors that support Intel 64 architecture: set to an undefined but canonical value.
      - Bits 63:32 of the base addresses for SS, DS, and ES on processors that support Intel 64 architecture: cleared to 0.

GDTR and IDTR are loaded using the base and limit fields.

22.4 LOADING MSRS

VM entries may load MSRs from the VM-entry MSR-load area (see Section 20.8.2). Specifically each entry in that area (up to the number specified in the VM-entry MSR-load count) is processed in order by loading the MSR indexed by bits 31:0 with the contents of bits 127:64 as they would be written by WRMSR.[1]

Processing of an entry fails in any of the following cases:

- The value of bits 31:0 is either C0000100H (the IA32_FS_BASE MSR) or C0000101 (the IA32_GS_BASE MSR).
- The value of bits 31:8 is 000008H, meaning that the indexed MSR is one that allows access to an APIC register when the local APIC is in x2APIC mode.
- The value of bits 31:0 indicates an MSR that can be written only in system-management mode (SMM) and the VM entry did not commence in SMM. (IA32_SMM_MONITOR_CTL is an MSR that can be written only in SMM.)
- The value of bits 31:0 indicates an MSR that cannot be loaded on VM entries for model-specific reasons. A processor may prevent loading of certain MSRs even if they can normally be written by WRMSR. Such model-specific behavior is documented in Appendix B.
- Bits 63:32 are not all 0.
- An attempt to write bits 127:64 to the MSR indexed by bits 31:0 of the entry would cause a general-protection exception if executed via WRMSR with CPL = 0.[2]

The VM entry fails if processing fails for any entry. The logical processor responds to such failures by loading state from the host-state area, as it would for a VM exit. See Section 22.7.

If any MSR is being loaded in such a way that would architecturally require a TLB flush, the TLBs are updated so that, after VM entry, the logical processor will not use any translations that were cached before the transition.

1. Because attempts to modify the value of IA32_EFER.LMA by WRMSR are ignored, attempts to modify it using the VM-entry MSR-load area are also ignored.
2. If CR0.PG = 1, WRMSR to the IA32_EFER MSR causes a general-protection exception if it would modify the LME bit. If VM entry has established CR0.PG = 1, the IA32_EFER MSR should not be included in the VM-entry MSR-load area for the purpose of modifying the LME bit.

22.5.1.2 VM Exits During Event Injection

An event being injected never causes a VM exit directly regardless of the settings of the VM-execution controls. For example, setting the "NMI exiting" VM-execution control to 1 does not cause a VM exit due to injection of an NMI.

However, the event-delivery process may lead to a VM exit:

- If the vector in the VM-entry interruption-information field identifies a task gate in the IDT, the attempted task switch may cause a VM exit just as it would had the injected event occurred during normal execution in VMX non-root operation (see Section 21.6.2).
- If event delivery encounters a nested exception, a VM exit may occur depending on the contents of the exception bitmap (see Section 21.3).
- If event delivery generates a double-fault exception (due to a nested exception); the logical processor encounters another nested exception while attempting to call the double-fault handler; and that exception does not cause a VM exit due to the exception bitmap; then a VM exit occurs due to triple fault (see Section 21.3).
- If event delivery injects a double-fault exception and encounters a nested exception that does not cause a VM exit due to the exception bitmap, then a VM exit occurs due to triple fault (see Section 21.3).
- If the "virtualize APIC accesses" VM-execution control is 1 and event delivery generates an access to the APIC-access page, that access may cause an APIC-access VM exit (see Section 21.2) or, if the access is a VTPR access, be treated as specified in Section 21.5.3.[1]

1. "Virtualize APIC accesses" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "virtualize APIC accesses" VM-execution control were 0. See Section 20.6.2.

22.5.1.3 Event Injection for VM Entries to Real-Address Mode

If VM entry is loading CR0.PE with 0, any injected vectored event is delivered as would normally be done in real-address mode.[1] Specifically, VM entry uses the vector provided in the VM-entry interruption-information field to select a 4-byte entry from an interrupt-vector table at the linear address in IDTR.base. Further details are provided in Section 15.1.4 in Volume 3A of the *IA-32 Intel® Architecture Software Developer's Manual*.

Because bit 11 (deliver error code) in the VM-entry interruption-information field must be 0 if CR0.PE will be 0 after VM entry (see Section 22.2.1.3), vectored events injected with CR0.PE = 0 do not push an error code on the stack. This is consistent with event delivery in real-address mode.

If event delivery encounters a fault (due to a violation of IDTR.limit or of SS.limit), the fault is treated as if it had occurred during event delivery in VMX non-root operation. Such a fault may lead to a VM exit as discussed in Section 22.5.1.2.

22.6.4 VMX-Preemption Timer

If the "activate VMX-preemption timer" VM-execution control is 1, VM entry starts the VMX-preemption timer with the unsigned value in the VMX-preemption timer-value field.

If the "activate VMX-preemption timer" 1 and the value in the VMX-preemption timer-value field is zero, a VM exit occurs before execution of any instruction following VM entry (if it is not blocked by activity state). The VM exit occurs with its normal priority after any event injection. For example, any pending debug exceptions established by VM entry (see Section 22.6.3) take priority over a timer-induced VM exit. (The timer-induced VM exit will occur after delivery of the debug exception, unless that exception or its delivery causes a different VM exit.)

See Section 21.7.1 for details of the operation of the VMX-preemption timer in VMX non-root operation, including the blocking and priority of the VM exits that it causes.

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation, VM entry must be loading CR0.PE with 1 unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

22.6.7 VM Exits Induced by the TPR Shadow

If the "use TPR shadow" and "virtualize APIC accesses" VM-execution controls are both 1, a VM exit occurs immediately after VM entry if the value of bits 3:0 of the TPR threshold VM-execution control field is greater than the value of bits 7:4 in byte 80H on the virtual-APIC page (see Section 20.6.8).[1]

The following items detail the treatment of these VM exits:

- The VM exits are not blocked if RFLAGS.IF = 0 or by the setting of bits in the interruptibility-state field in guest-state area.
- The VM exits follow event injection if such injection is specified for VM entry.
- VM exits caused by this control take priority over system-management interrupts (SMIs), INIT signals, and lower priority events. They thus have priority over the VM exits described in Section 22.6.5, Section 22.6.6, and Section 22.6.8, as well as any interrupts or debug exceptions that may be pending at the time of VM entry.
- These VM exits wake the logical processor if it just entered the HLT state as part of a VM entry (see Section 22.6.2). They do not occur if the logical processor just entered the shutdown state or the wait-for-SIPI state.

  If such a VM exit is suppressed because the processor just entered the shutdown state, it occurs after the delivery of any event that cause the logical processor to leave the shutdown state while remaining in VMX non-root operation (e.g., due to an NMI that occurs while the "NMI-exiting" VM-execution control is 0).
- The basic exit reason is "TPR below threshold."

22.7 VM-ENTRY FAILURES DURING OR AFTER LOADING GUEST STATE

VM-entry failures due to the checks identified in Section 22.3.1 and failures during the MSR loading identified in Section 22.4 are treated differently from those that occur earlier in VM entry. In these cases, the following steps take place:

1. Information about the VM-entry failure is recorded in the VM-exit information fields:
   - Exit reason.
     - Bits 15:0 of this field contain the basic exit reason. It is loaded with a number indicating the general cause of the VM-entry failure. The following numbers are used:

       33. VM-entry failure due to invalid guest state. A VM entry failed one of the checks identified in Section 22.3.1.

       34. VM-entry failure due to MSR loading. A VM entry failed in an attempt to load MSRs (see Section 22.4).

       41. VM-entry failure due to machine check. A machine check occurred during VM entry (see Section 22.8).
     - Bit 31 is set to 1 to indicate a VM-entry failure.

1. "Virtualize APIC accesses" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "virtualize APIC accesses" VM-execution control were 0. See Section 20.6.2.

- The remainder of the field (bits 30:16) is cleared.

— Exit qualification. This field is set based on the exit reason.

- VM-entry failure due to invalid guest state. In most cases, the exit qualification is cleared to 0. The following non-zero values are used in the cases indicated:

  1. Not used.
  2. Failure was due to a problem loading the PDPTEs (see Section 22.3.1.6).
  3. Failure was due to an attempt to inject a non-maskable interrupt (NMI) into a guest that is blocking events through the STI blocking bit in the interruptibility-state field. Such failures are implementation-specific (see Section 22.3.1.5).

  4.Failure was due to an invalid VMCS link pointer (see Section 22.3.1.5).

  VM-entry checks on guest-state fields may be performed in any order. Thus, an indication by exit qualification of one cause does not imply that there are not also other errors. Different processors may give different exit qualifications for the same VMCS.

- VM-entry failure due to MSR loading. The exit qualification is loaded to indicate which entry in the VM-entry MSR-load area caused the problem (1 for the first entry, 2 for the second, etc.).

— All other VM-exit information fields are unmodified.

2. Processor state is loaded as would be done on a VM exit (see Section 23.5). If this results in [CR4.PAE & CR0.PG & ~IA32_EFER.LMA] = 1, page-directory-pointer-table entries (PDPTEs) may be checked and loaded (see Section 23.5.4).
3. The state of blocking by NMI is what it was before VM entry.
4. MSRs are loaded as specified in the VM-exit MSR-load area (see Section 23.6).

Although this process resembles that of a VM exit, many steps taken during a VM exit do not occur for these VM-entry failures:

- Most VM-exit information fields are not updated (see step 1 above).
- The valid bit in the VM-entry interruption-information field is not cleared.
- The guest-state area is not modified.
- No MSRs are saved into the VM-exit MSR-store area.

15. Real-mode Virtualization Updates to Chapter 23 (Chapter 23, Vol 3B)

Change bars show changes to Chapter 23 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

23.2 RECORDING VM-EXIT INFORMATION AND UPDATING VM-ENTRY CONTROL FIELDS

VM exits begin by recording information about the nature of and reason for the VM exit in the VM-exit information fields. Section 23.2.1 to Section 23.2.4 detail the use of these fields.

In addition to updating the VM-exit information fields, the valid bit (bit 31) is cleared in the VM-entry interruption-information field. If bit 5 of the IA32_VMX_MISC MSR (index 485H) is read as 1 (see Appendix G.6), the value of IA32_EFER.LMA is stored into the "IA-32e mode guest" VM-entry control.[1]

23.2.1 Basic VM-Exit Information

Section 20.9.1 defines the basic VM-exit information fields. The following items detail their use.

- **Exit reason.**
  - Bits 15:0 of this field contain the basic exit reason. It is loaded with a number indicating the general cause of the VM exit. Appendix I lists the numbers used and their meaning.
  - The remainder of the field (bits 31:16) is cleared on every VM exit.
- **Exit qualification.** This field is saved for VM exits due to the following causes: debug exceptions; page-fault exceptions; start-up IPIs (SIPIs); system-management interrupts (SMIs) that arrive immediately after the retirement of I/O instructions; task switches; INVEPT; INVLPG;INVVPID; LGDT; LIDT; LLDT; LTR; SGDT; SIDT; SLDT; STR; VMCLEAR; VMPTRLD; VMPTRST; VMREAD; VMWRITE; VMXON; control-register accesses; MOV DR; I/O instructions; MWAIT; accesses to the APIC-access page (see Section 21.2); and EPT violations. For all other VM exits, this field is cleared. The following items provide details:
  - For a debug exception, the exit qualification contains information about the debug exception. The information has the format given in Table 23-1.

Table 23-1 Exit Qualification for Debug Exceptions

Bit Position(s)	Contents
3:0	B3 – B0. When set, each of these bits indicates that the corresponding breakpoint condition was met. Any of these bits may be set even if its corresponding enabling bit in DR7 is not set.
12:4	Reserved (cleared to 0).

1. Bit 5 of the IA32_VMX_MISC MSR is read as 1 on any logical processor that supports the 1-setting of the "unrestricted guest" VM-execution control.

Table 23-1 Exit Qualification for Debug Exceptions (Continued)

Bit Position(s)	Contents
13	BD. When set, this bit indicates that the cause of the debug exception is "debug register access detected."
14	BS. When set, this bit indicates that the cause of the debug exception is either the execution of a single instruction (if RFLAGS.TF = 1 and IA32_DEBUGCTL.BTF = 0) or a taken branch (if RFLAGS.TF = DEBUGCTL.BTF = 1).
63:15	Reserved (cleared to 0). Bits 63:32 exist only on processors that support Intel 64 architecture.

— For a page-fault exception, the exit qualification contains the linear address that caused the page fault. On processors that support Intel 64 architecture, bits 63:32 are cleared if the logical processor was not in 64-bit mode before the VM exit.

— For a start-up IPI (SIPI), the exit qualification contains the SIPI vector information in bits 7:0. Bits 63:8 of the exit qualification are cleared to 0.

— For a task switch, the exit qualification contains details about the task switch, encoded as shown in Table 23-2.

Table 23-2 Exit Qualification for Task Switch

Bit Position(s)	Contents
15:0	Selector of task-state segment (TSS) to which the guest attempted to switch
29:16	Reserved (cleared to 0)
31:30	Source of task switch initiation: 0: CALL instruction 1: IRET instruction 2: JMP instruction 3: Task gate in IDT
63:32	Reserved (cleared to 0). These bits exist only on processors that support Intel 64 architecture.

— For INVLPG, the exit qualification contains the linear-address operand of the instruction.

- On processors that support Intel 64 architecture, bits 63:32 are cleared if the logical processor was not in 64-bit mode before the VM exit.
- If the INVLPG source operand specifies an unusable segment, the linear address specified in the exit qualification will match the linear address that the INVLPG would have used if no VM exit occurred. This address is not architecturally defined and may be implementation-specific.

— For INVEPT, INVVPID, LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, STR, VMCLEAR, VMPTRLD, VMPTRST, VMREAD, VMWRITE, and VMXON, the exit qualification receives the value of the instruction's displacement field, which is sign-extended to 64 bits if necessary (32 bits on processors that do not support Intel 64 archi-

tecture). If the instruction has no displacement (for example, has a register operand), zero is stored into the exit qualification.

On processors that support Intel 64 architecture, an exception is made for RIP-relative addressing (used only in 64-bit mode). Such addressing causes an instruction to use an address that is the sum of the displacement field and the value of RIP that references the following instruction. In this case, the exit qualification is loaded with the sum of the displacement field and the appropriate RIP value.

In all cases, bits of this field beyond the instruction's address size are undefined. For example, suppose that the address-size field in the VM-exit instruction-information field (see Section 20.9.4 and Section 23.2.4) reports an *n*-bit address size. Then bits 63:*n* (bits 31:*n* on processors that do not support Intel 64 architecture) of the instruction displacement are undefined.

23.2.2 Information for VM Exits Due to Vectored Events

Section 20.9.2 defines fields containing information for VM exits due to the following events: exceptions (including those generated by the instructions INT3, INTO, BOUND, and UD2); external interrupts that occur while the "acknowledge interrupt on exit" VM-exit control is 1; and non-maskable interrupts (NMIs). Such VM exits include those that occur on an attempt at a task switch that causes an exception before generating the VM exit due to the task switch that causes the VM exit.

The following items detail the use of these fields:

- **VM-exit interruption information** (format given in Table 20-14). The following items detail how this field is established for VM exits due to these events:
  - For an exception, bits 7:0 receive the exception vector (at most 31). For an NMI, bits 7:0 are set to 2. For an external interrupt, bits 7:0 receive the interrupt number.
  - Bits 10:8 are set to 0 (external interrupt), 2 (non-maskable interrupt), 3 (hardware exception), or 6 (software exception). Hardware exceptions comprise all exceptions except breakpoint exceptions (#BP; generated by INT3) and overflow exceptions (#OF; generated by INTO); these are software exceptions. BOUND-range exceeded exceptions (#BR; generated by BOUND) and invalid opcode exceptions (#UD) generated by UD2 are hardware exceptions.
  - Bit 11 is set to 1 if the VM exit is caused by a hardware exception that would have delivered an error code on the stack. This bit is always 0 if the VM exit occurred while the logical processor was in real-address mode (CR0.PE=0).[1] If bit 11 is set to 1, the error code is placed in the VM-exit interruption error code (see below).
  - Bit 12 is undefined in any of the following cases:
    - If the "NMI exiting" VM-execution control is 1 and the "virtual NMIs" VM-execution control is 0.
    - If the VM exit sets the valid bit in the IDT-vectoring information field (see Section 23.2.3).

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation, a logical processor cannot be in real-address mode unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

- If the VM exit is due to a double fault (the interruption type is hardware exception and the vector is 8).

Otherwise, bit 12 is defined as follows:

- If the "virtual NMIs" VM-execution control is 0, the VM exit is due to a fault on the IRET instruction (other than a debug exception for an instruction breakpoint), and blocking by NMI (see Table 20-3) was in effect before execution of IRET, bit 12 is set to 1.
- If the "virtual NMIs" VM-execution control is 1, the VM exit is due to a fault on the IRET instruction (other than a debug exception for an instruction breakpoint), and virtual-NMI blocking was in effect before execution of IRET, bit 12 is set to 1.
- For all other relevant VM exits, bit 12 is cleared to 0.[1]

— Bits 30:13 are always set to 0.

— Bit 31 is always set to 1.

For other VM exits (including those due to external interrupts when the "acknowledge interrupt on exit" VM-exit control is 0), the field is marked invalid (by clearing bit 31) and the remainder of the field is undefined.

- VM-exit interruption error code.

— For VM exits that set both bit 31 (valid) and bit 11 (error code valid) in the VM-exit interruption-information field, this field receives the error code that would have been pushed on the stack had the event causing the VM exit been delivered normally through the IDT. The EXT bit is set in this field exactly when it would be set normally. For exceptions that occur during the delivery of double fault (if the IDT-vectoring information field indicates a double fault), the EXT bit is set to 1, assuming that (1) that the exception would produce an error code normally (if not incident to double-fault delivery) and (2) that the error code uses the EXT bit (not for page faults, which use a different format).

— For other VM exits, the value of this field is undefined.

23.2.3 Information for VM Exits During Event Delivery

Section 20.9.3 defined fields containing information for VM exits that occur while delivering an event through the IDT and as a result of any of the following cases:[2]

- A fault occurs during event delivery and causes a VM exit (because the bit associated with the fault is set to 1 in the exception bitmap).
- A task switch is invoked through a task gate in the IDT. The VM exit occurs due to the task switch only after the initial checks of the task switch pass (see Section 21.6.2).
- Event delivery causes an APIC-access VM exit (see Section 21.2).
- An EPT violation or EPT misconfiguration that occurs during event delivery.

These fields are used for VM exits that occur during delivery of events injected as part of VM entry (see Section 22.5.1.2).

1. The conditions imply that, if the "NMI exiting" VM-execution control is 0 or the "virtual NMIs" VM-execution control is 1, bit 12 is always cleared to 0 by VM exits due to debug exceptions.
2. This includes the case in which a VM exit occurs while delivering a software interrupt (INT *n*) through the 16-bit IVT (interrupt vector table) that is used in virtual-8086 mode with virtual-machine extensions (if RFLAGS.VM = CR4.VME = 1).

A VM exit is not considered to occur during event delivery in any of the following circumstances:

- The original event causes the VM exit directly (for example, because the original event is a non-maskable interrupt (NMI) and the "NMI exiting" VM-execution control is 1).
- The original event results in a double-fault exception that causes the VM exit directly.
- The VM exit occurred as a result of fetching the first instruction of the handler invoked by the event delivery.
- The VM exit is caused by a triple fault.

The following items detail the use of these fields:

- IDT-vectoring information (format given in Table 20-15). The following items detail how this field is established for VM exits that occur during event delivery:
  - If the VM exit occurred during delivery of an exception, bits 7:0 receive the exception vector (at most 31). If the VM exit occurred during delivery of an NMI, bits 7:0 are set to 2. If the VM exit occurred during delivery of an external interrupt, bits 7:0 receive the interrupt number.
  - Bits 10:8 are set to indicate the type of event that was being delivered when the VM exit occurred: 0 (external interrupt), 2 (non-maskable interrupt), 3 (hardware exception), 4 (software interrupt), 5 (privileged software interrupt), or 6 (software exception).

    Hardware exceptions comprise all exceptions except breakpoint exceptions (#BP; generated by INT3) and overflow exceptions (#OF; generated by INTO); these are software exceptions. BOUND-range exceeded exceptions (#BR; generated by BOUND) and invalid opcode exceptions (#UD) generated by UD2 are hardware exceptions.

    Bits 10:8 may indicate privileged software interrupt if such an event was injected as part of VM entry.
  - Bit 11 is set to 1 if the VM exit occurred during delivery of a hardware exception that would have delivered an error code on the stack. This bit is always 0 if the VM exit occurred while the logical processor was in real-address mode (CR0.PE=0).[1] If bit 11 is set to 1, the error code is placed in the IDT-vectoring error code (see below).
  - Bit 12 is undefined.
  - Bits 30:13 are always set to 0.
  - Bit 31 is always set to 1.

  For other VM exits, the field is marked invalid (by clearing bit 31) and the remainder of the field is undefined.
- IDT-vectoring error code.
  - For VM exits that set both bit 31 (valid) and bit 11 (error code valid) in the IDT-vectoring information field, this field receives the error code that would have been pushed on the stack by the event that was being delivered through the IDT at the time of the VM exit. The EXT bit is set in this field when it would be set normally.

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PE must be 1 in VMX operation, a logical processor cannot be in real-address mode unless the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

— For other VM exits, the value of this field is undefined.

23.2.4 Information for VM Exits Due to Instruction Execution

Section 20.9.4 defined fields containing information for VM exits that occur due to instruction execution. (The VM-exit instruction length is also used for VM exits that occur during the delivery of a software interrupt or software exception.) The following items detail their use.

- **VM-exit instruction length.** This field is used in the following cases:
  - — For fault-like VM exits due to attempts to execute one of the following instructions that cause VM exits unconditionally (see Section 21.1.2) or based on the settings of VM-execution controls (see Section 21.1.3): CLTS, CPUID, GETSEC, HLT, IN, INS, INVD, INVEPT, INVLPG, INVVPID, LGDT, LIDT, LLDT, LMSW, LTR, MONITOR, MOV CR, MOV DR, MWAIT, OUT, OUTS, PAUSE, RDMSR, RDPMC, RDTSC, RDTSCP, RSM, SGDT, SIDT, SLDT, STR, VMCALL, VMCLEAR, VMLAUNCH, VMPTRLD, VMPTRST, VMREAD, VMRESUME, VMWRITE, VMXOFF, VMXON, WBINVD, WRMSR, and XSETBV.[1]
  - — For VM exits due to software exceptions (those generated by executions of INT3 or INTO).
  - — For VM exits due to faults encountered during delivery of a software interrupt, privileged software exception, or software exception.
  - — For VM exits due to attempts to effect a task switch via instruction execution. These are VM exits that produce an exit reason indicating task switch and either of the following:
    - An exit qualification indicating execution of CALL, IRET, or JMP instruction.
    - An exit qualification indicating a task gate in the IDT and an IDT-vectoring information field indicating that the task gate was encountered during delivery of a software interrupt, privileged software exception, or software exception.
  - — For APIC-access VM exits resulting from linear accesses (see Section 21.2.1) and encountered during delivery of a software interrupt, privileged software exception, or software exception.[2]

  In all the above cases, this field receives the length in bytes (1–15) of the instruction (including any instruction prefixes) whose execution led to the VM exit (see the next paragraph for one exception).

  The cases of VM exits encountered during delivery of a software interrupt, privileged software exception, or software exception include those encountered during delivery of events injected as part of VM entry (see Section 22.5.1.2). If the original event was injected as part of VM entry, this field receives the value of the VM-entry instruction length.

1. This item applies only to fault-like VM exits. It does not apply to trap-like VM exits following executions of the MOV to CR8 instruction when the "use TPR shadow" VM-execution control is 1 or to those following executions of the WRMSR instruction when the "virtualize x2APIC mode" VM-execution control is 1.
2. The VM-exit instruction-length field is not defined following APIC-access VM exits resulting from physical accesses (see Section 21.2.3) even if encountered during delivery of a software interrupt, privileged software exception, or software exception.

All VM exits other than those listed in the above items leave this field undefined.

- **VM-exit instruction information**. For VM exits due to attempts to execute INS, INVEPT, INVVPID, LIDT, LGDT, LLDT, LTR, OUTS, SIDT, SGDT, SLDT, STR, VMCLEAR, VMPTRLD, VMPTRST, VMREAD, VMWRITE, or VMXON, this field receives information about the instruction that caused the VM exit. The format of the field depends on the identity of the instruction causing the VM exit:
  - For VM exits due to attempts to execute INS or OUTS, the field has the format is given in Table 23-8.[1]

Table 23-8 Format of the VM-Exit Instruction-Information Field as Used for INS and OUTS

Bit Position(s)	Content
6:0	Undefined.
9:7	Address size: 0: 16-bit 1: 32-bit 2: 64-bit (used only on processors that support Intel 64 architecture) Other values not used.
14:10	Undefined.
17:15	Segment register: 0: ES 1: CS 2: SS 3: DS 4: FS 5: GS Other values not used. Undefined for VM exits due to execution of INS.
31:18	Undefined.

23.3.1 Saving Control Registers, Debug Registers, and MSRs

Contents of certain control registers, debug registers, and MSRs is saved as follows:

- The contents of CR0, CR3, CR4, and the IA32_SYSENTER_CS, IA32_SYSENTER_ESP, and IA32_SYSENTER_EIP MSRs are saved into the corresponding fields. Bits 63:32 of the IA32_SYSENTER_CS MSR are not saved. On processors that do not support Intel 64 architecture, bits 63:32 of the IA32_SYSENTER_ESP and IA32_SYSENTER_EIP MSRs are not saved.
- If the "save debug controls" VM-exit control is 1, the contents of DR7 and the IA32_DEBUGCTL MSR are saved into the corresponding fields. The first processors to support the virtual-machine extensions supported only the 1-setting of this control and thus always saved data into these fields.

1. The format of the field was undefined for these VM exits on the first processors to support the virtual-machine extensions. Software can determine whether the format specified in Table 23-8 is used by consulting the VMX capability MSR IA32_VMX_BASIC (see Appendix G.1).

- If the "save IA32_PAT" VM-exit control is 1, the contents of the IA32_PAT MSR are saved into the corresponding field.
- If the "save IA32_EFER" VM-exit control is 1, the contents of the IA32_EFER MSR are saved into the corresponding field.
- The value of the SMBASE field is undefined after all VM exits except SMM VM exits. See Section 25.15.2.

23.3.4 Saving Non-Register State

Information corresponding to guest non-register state is saved as follows:

- The activity-state field is saved with the logical processor's activity state before the VM exit.[1] See Section 23.1 for details of how events leading to a VM exit may affect the activity state.
- The interruptibility-state field is saved to reflect the logical processor's interruptibility before the VM exit. See Section 23.1 for details of how events leading to a VM exit may affect this state. VM exits that end outside system-management mode (SMM) save bit 2 (blocking by SMI) as 0 regardless of the state of such blocking before the VM exit.

  Bit 3 (blocking by NMI) is treated specially if the "virtual NMIs" VM-execution control is 1. In this case, the value saved for this field does not indicate the blocking of NMIs but rather the state of virtual-NMI blocking.
- The pending debug exceptions field is saved as clear for all VM exits except the following:
  - A VM exit caused by an INIT signal, a machine-check exception, or a system-management interrupt (SMI).
  - A VM exit with basic exit reason either "TPR below threshold."[2]
  - A VM exit with basic exit reason "monitor trap flag."
  - VM exits that are not caused by debug exceptions and that occur while there is MOV-SS blocking of debug exceptions.

  For VM exits that do not clear the field, the value saved is determined as follows:
  - Each of bits 3:0 may be set if it corresponds to a matched breakpoint. This may be true even if the corresponding breakpoint is not enabled in DR7.
  - Suppose that a VM exit is due to an INIT signal, a machine-check exception, or an SMI; or that a VM exit has basic exit reason "TPR below threshold" or "monitor trap flag." In this case, the value saved sets bits corresponding to the causes of any debug exceptions that were pending at the time of the VM exit.

    If the VM exit occurs immediately after VM entry, the value saved may match that which was loaded on VM entry (see Section 22.6.3). Otherwise, the following items apply:
    - Bit 12 (enabled breakpoint) is set to 1 if there was at least one matched data or I/O breakpoint that was enabled in DR7. Bit 12 is also set if it had been set on VM entry, causing there to be valid pending debug exceptions (see Section

1. If this activity state was an inactive state resulting from execution of a specific instruction (HLT or MWAIT), the value saved for RIP by that VM exit will reference the following instruction.
2. This item includes VM exits that occur after executions of MOV to CR8 or WRMSR (Section 21.1.3), TPR-shadow updates (Section 21.5.3.3), and certain VM entries (Section 22.6.7).

22.6.3) and the VM exit occurred before those exceptions were either delivered or lost. In other cases, bit 12 is cleared to 0.

- Bit 14 (BS) is set if RFLAGS.TF = 1 in either of the following cases:
  - IA32_DEBUGCTL.BTF = 0 and the cause of a pending debug exception was the execution of a single instruction.
  - IA32_DEBUGCTL.BTF = 1 and the cause of a pending debug exception was a taken branch.

— Suppose that a VM exit is due to another reason (but not a debug exception) and occurs while there is MOV-SS blocking of debug exceptions. In this case, the value saved sets bits corresponding to the causes of any debug exceptions that were pending at the time of the VM exit. If the VM exit occurs immediately after VM entry (no instructions were executed in VMX non-root operation), the value saved may match that which was loaded on VM entry (see Section 22.6.3). Otherwise, the following items apply:

- Bit 12 (enabled breakpoint) is set to 1 if there was at least one matched data or I/O breakpoint that was enabled in DR7. Bit 12 is also set if it had been set on VM entry, causing there to be valid pending debug exceptions (see Section 22.6.3) and the VM exit occurred before those exceptions were either delivered or lost. In other cases, bit 12 is cleared to 0.
- The setting of bit 14 (BS) is implementation-specific. However, it is not set if RFLAGS.TF = 0 or IA32_DEBUGCTL.BTF = 1.

— The reserved bits in the field are cleared.

- If the "save VMX-preemption timer value" VM-exit control is 1, the value of timer is saved into the VMX-preemption timer-value field. This is the value loaded from this field on VM entry as subsequently decremented (see Section 21.7.1). (If the "save VMX-preemption timer value" VM-exit control is 0, VM exit does not modify the value of the VMX-preemption timer-value field.)
- If the logical processor supports the 1-setting of the "enable EPT" VM-execution control, values are saved into the four (4) PDPTE fields as follows:

  — If the "enable EPT" VM-execution control is 1 and the logical processor was using PAE paging at the time of the VM exit, the PDPTE values currently in use are saved:[1]

  - The values saved into bits 11:9 of each of the fields is undefined.
  - If the value saved into one of the fields has bit 0 (present) clear, the value saved into bits 63:1 of that field is undefined. That value need not correspond to the value that was loaded by VM entry or to any value that might have been loaded in VMX non-root operation.
  - If the value saved into one of the fields has bit 0 (present) set, the value saved into bits 63:12 of the field is a guest-physical address.

  — If the "enable EPT" VM-execution control is 0 or the logical processor was not using PAE paging at the time of the VM exit, the values saved are undefined.

1. A logical processor uses PAE paging if CR0.PG = 1, CR4.PAE = 1 and IA32_EFER.LMA = 0. See Section 3.8 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM exit functions as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.

23.5.1 Loading Host Control Registers, Debug Registers, MSRs

VM exits load new values for controls registers, debug registers, and some MSRs:

- CR0, CR3, and CR4 are loaded from the CR0 field, the CR3 field, and the CR4 field, respectively, with the following exceptions:
  - The following bits are not modified:
    - For CR0, ET, CD, NW; bits 63:32 (on processors that support Intel 64 architecture), 28:19, 17, and 15:6; and any bits that are fixed in VMX operation (see Section 19.8).[1]
    - For CR3, bits 63:52 and bits in the range 51:32 beyond the processor's physical-address width (they are cleared to 0).[2] (This item applies only to processors that support Intel 64 architecture.)
    - For CR4, any bits that are fixed in VMX operation (see Section 19.8).
  - CR4.PAE is set to 1 if the "host address-space size" VM-exit control is 1.
- DR7 is set to 400H.
- The following MSRs are established as follows:
  - The IA32_DEBUGCTL MSR is cleared to 00000000_00000000H.
  - The IA32_SYSENTER_CS MSR is loaded from the IA32_SYSENTER_CS field. Since that field has only 32 bits, bits 63:32 of the MSR are cleared to 0.
  - IA32_SYSENTER_ESP MSR and IA32_SYSENTER_EIP MSR are loaded from the IA32_SYSENTER_ESP field and the IA32_SYSENTER_EIP field, respectively. On processors that do not support Intel 64 architecture, these fields have only 32 bits; bits 63:32 of the MSRs are cleared to 0.

The following are performed on processors that support Intel 64 architecture:

- The MSRs FS.base and GS.base are loaded from the base-address fields for FS and GS, respectively (see Section 23.5.2).
- The LMA and LME bits in the IA32_EFER MSR are each loaded with the setting of the "host address-space size" VM-exit control.

  - If the "load IA32_PERF_GLOBAL_CTRL" VM-exit control is 1, the IA32_PERF_GLOBAL_CTRL MSR is loaded from the IA32_PERF_GLOBAL_CTRL field.
  - If the "load IA32_PAT" VM-exit control is 1, the IA32_PAT MSR is loaded from the IA32_PAT field.
  - If the "load IA32_EFER" VM-exit control is 1, the IA32_EFER MSR is loaded from the IA32_EFER field.

  With the exception of FS.base and GS.base, any of these MSRs is subsequently overwritten if it appears in the VM-exit MSR-load area. See Section 23.6.

1. Bits 28:19, 17, and 15:6 of CR0 and CR0.ET are unchanged by executions of MOV to CR0. CR0.ET is always 1 and the other bits are always 0.
2. Software can determine a processor's physical-address width by executing CPUID with 80000008H in EAX. The physical-address width is returned in bits 7:0 of EAX.

23.5.2 Loading Host Segment and Descriptor-Table Registers

Each of the registers CS, SS, DS, ES, FS, GS, and TR is loaded as follows (see below for the treatment of LDTR):

- The selector is loaded from the selector field. The segment is unusable if its selector is loaded with zero. The checks specified Section 22.3.1.2 limit the selector values that may be loaded. In particular, CS and TR are never loaded with zero and are thus never unusable. SS can be loaded with zero only on processors that support Intel 64 architecture and only if the VM exit is to 64-bit mode (64-bit mode allows use of segments marked unusable).
- The base address is set as follows:
  - CS. Cleared to zero.
  - SS, DS, and ES. Undefined if the segment is unusable; otherwise, cleared to zero.
  - FS and GS. Undefined (but, on processors that support Intel 64 architecture, canonical) if the segment is unusable and the VM exit is not to 64-bit mode; otherwise, loaded from the base-address field. On processors that support Intel 64 architecture, the values loaded for base addresses for FS and GS are also manifest in the FS.base and GS.base MSRs.
  - TR. Loaded from the host-state area.
- The segment limit is set as follows:
  - CS. Set to FFFFFFFFH (corresponding to a descriptor limit of FFFFFH and a G-bit setting of 1).
  - SS, DS, ES, FS, and GS. Undefined if the segment is unusable; otherwise, set to FFFFFFFFH.
  - TR. Set to 00000067H.
- The type field and S bit are set as follows:
  - CS. Type set to 11 and S set to 1 (execute/read, accessed, non-conforming code segment).
  - SS, DS, ES, FS, and GS. Undefined if the segment is unusable; otherwise, type set to 3 and S set to 1 (read/write, accessed, expand-up data segment).
  - TR. Type set to 11 and S set to 0 (busy 32-bit task-state segment).
- The DPL is set as follows:
  - CS, SS, and TR. Set to 0. The current privilege level (CPL) will be 0 after the VM exit completes.
  - DS, ES, FS, and GS. Undefined if the segment is unusable; otherwise, set to 0.
- The P bit is set as follows:
  - CS, TR. Set to 1.
  - SS, DS, ES, FS, and GS. Undefined if the segment is unusable; otherwise, set to 1.
- On processors that support Intel 64 architecture, CS.L is loaded with the setting of the “host address-space size” VM-exit control. Because the value of this control is also loaded into IA32_EFER.LMA (see Section 23.5.1), no VM exit is ever to compatibility mode (which requires IA32_EFER.LMA = 1 and CS.L = 0).
- D/B.

  - CS. Loaded with the inverse of the setting of the “host address-space size” VM-exit control. For example, if that control is 0, indicating a 32-bit guest, CS.D/B is set to 1.
  - SS, DS, ES, FS, and GS. Undefined if the segment is unusable; otherwise, set to 1.
  - TR. Set to 0.
- G.
  - CS. Set to 1.
  - SS, DS, ES, FS, and GS. Undefined if the segment is unusable; otherwise, set to 1.
  - TR. Set to 0.

The host-state area does not contain a selector field for LDTR. LDTR is established as follows on all VM exits: the selector is cleared to 0000H, the segment is marked unusable and is otherwise undefined (although the base address is always canonical).

The base addresses for GDTR and IDTR are loaded from the GDTR base-address field and the IDTR base-address field, respectively. The GDTR and IDTR limits are each set to FFFFH.

16. Real-mode Virtualization Updates to Chapter 24 (Chapter 24, Vol 3B)

Change bars show changes to Chapter 24 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

24.2.1 EPT Overview

EPT is used when the "enable EPT" VM-execution control is 1.[1] It translates the guest-physical addresses used in VMX non-root operation and those used by VM entry for event injection.

The translation from guest-physical addresses to physical addresses is determined by a set of **EPT paging structures**. The EPT paging structures are similar to those used to translate linear addresses while the processor is in IA-32e mode. Details of the EPT paging structures are given in Section 24.2.3.

If CR0.PG = 1, linear addresses are translated through paging structures referenced through control register CR3 . While the "enable EPT" VM-execution control is 1, these are called **guest paging structures**. There are no guest paging structures if CR0.PG = 0.[2]

When the "enable EPT" VM-execution control is 1, the identity of **guest-physical addresses** depends on the value of CR0.PG:

- If CR0.PG = 0, each linear address is treated as a guest-physical address.
- If CR0.PG = 1, guest-physical addresses are those derived from the contents of control register CR3 and the guest paging structures. (This includes the values of the PDPTEs, which logical processors store in internal, non-architectural registers.) The latter includes (in page-table entries and in page-directory entries for which bit 7—PS—is 1) the addresses to which linear addresses are translated by the guest paging structures.

If CR0.PG = 1, the translation of a linear address to a physical address requires multiple translations of guest-physical addresses using EPT. Assume, for example, that CR4.PAE = CR4.PSE = 0. The translation of a 32-bit linear address then operates as follows:

- Bits 31:22 of the linear address select an entry in the guest page directory located at the guest-physical address in CR3. The guest-physical address of the guest page-directory entry (PDE) is translated through EPT to determine the guest PDE's physical address.
- Bits 21:12 of the linear address select an entry in the guest page table located at the guest-physical address in the guest PDE. The guest-physical address of the guest page-table entry (PTE) is translated through EPT to determine the guest PTE's physical address.
- Bits 11:0 of the linear address is the offset in the page frame located at the guest-physical address in the guest PTE. The guest-physical address determined by this

1. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, the logical processor operates as if the "enable EPT" VM-execution control were 0. See Section 20.6.2.
2. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, CR0.PG can be 0 in VMX non-root operation only if the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

offset is translated through EPT to determine the physical address to which the original linear address translates.

In addition to translating a guest-physical address to a physical address, EPT specifies the privileges that software is allowed when accessing the address. Attempts at disallowed accesses are called **EPT violations** and cause VM exits. See Section 24.2.4.

A logical processor uses EPT to translate guest-physical addresses only when those addresses are used to access memory. This principle implies the following:

- The MOV to CR3 instruction loads CR3 with a guest-physical address. Whether that address is translated through EPT depends on whether PAE paging is being used.[1]
  - — If PAE paging is not being used, the instruction does not use that address to access memory and does **not** cause it to be translated through EPT. (If CR0.PG = 1, the address will be translated through EPT on the next memory accessing using a linear address.)
  - — If PAE paging is being used, the instruction loads the four (4) page-directory-pointer-table entries (PDPTEs) from that address and it **does** cause the address to be translated through EPT.
- The MOV to CR0 instruction establishes PAE paging if it results in CR0.PG = 1 and the following were held before the instruction executed: (1) CR0.PG = 0; (2) CR4.PAE = 1; and (3) IA32_EFER.LME = 0. Such an execution loads the PDPTEs from the guest-physical address in CR3. The address **is** translated through EPT.
- The MOV to CR4 instruction establishes PAE paging if it results in CR4.PAE = 1 and the following were held before the instruction executed: (1) CR0.PG = 1; (2) CR4.PAE = 0; and (3) IA32_EFER.LMA = 0. Such an execution loads the PDPTEs from the guest-physical address in CR3. The address **is** translated through EPT.
- The PDPTEs contain guest-physical addresses. The instructions that load the PDPTEs (see above) do not use those addresses to access memory and do **not** cause them to be translated through EPT. (The address in a PDPTE will be translated through EPT on the next memory accessing using a linear address that uses that PDPTE.)

24.2.3.1 EPT PML4 Entries

An EPT PML4 entry is identified using bits 47:39 of the guest-physical address (see Section 24.2.2) and thus controls access to a 512-Gbyte region of the guest-physical address space. Table 24-1 shows the format of an EPT PML4 entry.

Table 24-1 Format of an EPT PML4 Entry

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 512-GByte region controlled by this entry
1	Write access; indicates whether writes are allowed to the 512-GByte region controlled by this entry
2	Execute access; indicates whether instruction fetches are allowed from the 512-GByte region controlled by this entry

1. A logical processor uses PAE paging if CR0.PG = 1, CR4.PAE = 1 and IA32_EFER.LMA = 0. See Section 3.8 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*.

Table 24-1 Format of an EPT PML4 Entry (Continued)

Bit Position(s)	Contents
7:3	Reserved (must be 0)
11:8	Ignored
N–1:12	Physical address of 4-KByte aligned EPT page-directory-pointer table referenced by this entry[1]
51:N	Reserved (must be 0)
63:52	Ignored

NOTES:
1. N is the physical-address width supported by the processor. Software can determine a processor's physical-address width by executing CPUID with 80000008H in EAX. The physical-address width is returned in bits 7:0 of EAX.

If bits 2:0 of an EPT PML4 entry are all 0, the entry is considered to be "not present"; the logical processor ignores bits 63:3 of such an entry and will not use it to reference an EPT page-directory-pointer table.

24.2.3.2 EPT Page-Directory-Pointer-Table Entries

An EPT page-directory-pointer-table entry (PDPTE) is identified using bits 47:30 of the guest-physical address (see Section 24.2.2) and thus controls access to a 1-Gbyte region of the guest-physical address space. Table 24-2 shows the format of an EPT PDPTE.

Table 24-2 Format of an EPT Page-Directory-Pointer-Table Entry (PDPTE)

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 1-GByte region controlled by this entry
1	Write access; indicates whether writes are allowed to the 1-GByte region controlled by this entry
2	Execute access; indicates whether instruction fetches are allowed from the 1-GByte region controlled by this entry
7:3	Reserved (must be 0)
11:8	Ignored
N–1:12	Physical address of 4-KByte aligned EPT page directory referenced by this entry[1]
51:N	Reserved (must be 0)
63:52	Ignored

NOTES:
1. N is the physical-address width supported by the logical processor.

If bits 2:0 of an EPT PDPTE are all 0, the entry is considered to be "not present"; the logical processor ignores bits 63:3 of such an entry and will not use it to reference an EPT page directory.

24.2.3.3 EPT Page-Directory Entries

An EPT page-directory entry (PDE) is identified using bits 47:21 of the guest-physical address (see Section 24.2.2) and thus controls access to a 2-Mbyte region of the guest-physical address space. It may do so either by referencing an EPT page table or by mapping a single 2-Mbyte page; the value of bit 7 in the EPT PDE determines which mechanism is used.

If bit 7 of the EPT PDE is 0, the entry references an EPT page table. Table 24-3 shows the format of such an EPT PDE.

Table 24-3 Format of an EPT Page-Directory Entry that References an EPT Page Table

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 2-MByte region controlled by this entry
1	Write access; indicates whether writes are allowed to the 2-MByte region controlled by this entry
2	Execute access; indicates whether instruction fetches are allowed from the 2-MByte region controlled by this entry
6:3	Reserved (must be 0)
7	Must be 0 (otherwise, this entry maps a 2-MByte page)
11:8	Ignored
N–1:12	Physical address of 4-KByte aligned EPT page table referenced by this entry[1]
51:N	Reserved (must be 0)
63:52	Ignored

NOTES:
1. N is the physical-address width supported by the logical processor.

If bit 7 of the EPT PDE is 1, the entry maps a 2-MByte page. Table 24-4 shows the format of such an EPT PDE.

Table 24-4 Format of an EPT Page-Directory Entry that Maps a 2-MByte Page

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 2-MByte page referenced by this entry
1	Write access; indicates whether writes are allowed to the 2-MByte page referenced by this entry

Table 24-4 Format of an EPT Page-Directory Entry that Maps a 2-MByte Page

Bit Position(s)	Contents
2	Execute access; indicates whether instruction fetches are allowed from the 2-MByte page referenced by this entry
5:3	EPT memory type for this 2-MByte page (see Section 24.2.5)
6	Ignore PAT memory type for this 2-MByte page (see Section 24.2.5)
7	Must be 1 (otherwise, this entry references an EPT page table)
11:8	Ignored
20:12	Reserved (must be 0)
N–1:21	Physical address of the 2-MByte page referenced by this entry[1]
51:N	Reserved (must be 0)
63:52	Ignored

NOTES:
1. N is the physical-address width supported by the logical processor.

If bits 2:0 of an EPT PDE are all 0, the entry is considered to be "not present"; the logical processor ignores bits 63:3 (including bit 7) of such an entry and will use it neither to reference an EPT page table nor to map a 2-MByte page.

24.2.3.4 EPT Page-Table Entries

An EPT page-table entry (PTE) is identified using bits 47:12 of the guest-physical address (see Section 24.2.2) and thus maps a 4-Kbyte page. Table 24-5 shows the format of an EPT PDE.

Table 24-5 Format of an EPT Page-Table Entry

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 4-KByte page referenced by this entry
1	Write access; indicates whether writes are allowed to the 4-KByte page referenced by this entry
2	Execute access; indicates whether instruction fetches are allowed from the 4-KByte page referenced by this entry
5:3	EPT memory type for this 4-KByte page (see Section 24.2.5)
6	Ignore PAT memory type for this 4-KByte page (see Section 24.2.5)
11:7	Ignored

Table 24-5 Format of an EPT Page-Table Entry (Continued)

Bit Position(s)	Contents
N–1:12	Physical address of the 4-KByte page referenced by this entry[1]
51:N	Reserved (must be 0)
63:52	Ignored

NOTES:
1. N is the physical-address width supported by the logical processor.

If bits 2:0 of an EPT PTE are all 0, the entry is considered to be "not present"; the logical processor ignores bits 63:3 of such an entry and will not use it to map a 4-KByte page.

24.2.4 EPT-Induced VM Exits

Accesses using guest-physical addresses may cause VM exits due to **EPT misconfigurations** and **EPT violations**. An EPT misconfiguration occurs when, in the course of translation a guest-physical address, the logical processor encounters an EPT paging-structure entry that contains an unsupported value. An EPT violation occurs when there is no EPT misconfiguration but the EPT paging-structure entries disallow an access using the guest-physical address.

EPT misconfigurations and EPT violations occur only due to an attempt to access memory with a guest-physical address. Loading CR3 with a guest-physical address with the MOV to CR3 instruction can cause neither an EPT configuration nor an EPT violation until that address is used to access a paging structure.[1]

24.2.4.3 Prioritization of EPT-Induced VM Exits

The translation of a linear address to a physical address requires one or more translations of guest-physical addresses using EPT (see Section 24.2.1). This section specifies the relative priority of EPT-induced VM exits with respect to each other and to other events that may be encountered when accessing memory using a linear address.

For an access to a guest-physical address, determination of whether an EPT misconfiguration or an EPT violation occurs is based on an iterative process:[2]

1. An EPT paging-structure entry is read (initially, this is an EPT PML4 entry):
   a. If the entry is not present (bits 2:0 are all 0), an EPT violation occurs.
   b. If the entry is present but its contents are not configured properly (see Section 24.2.4.1), an EPT misconfiguration occurs.

---

1. If the logical processor is using PAE paging—because CR0.PG = CR4.PAE = 1 and IA32_EFER.LMA = 0—the MOV to CR3 instruction loads the PDPTEs from memory using the guest-physical address being loaded into CR3. In this case, therefore, the MOV to CR3 instruction may cause an EPT misconfiguration or an EPT violation.
2. This is a simplification of the more detailed description given in Section 24.2.2.

c. If the entry is present and its contents are configured properly, operation depends on whether the entry references another EPT paging structure (whether it is an EPT PDE with bit 7 set to 1 or an EPT PTE):

   i) If the entry does references another EPT paging structure, an entry from that structure is accessed; step 1 is executed for that other entry.

   ii) Otherwise, the entry is used to produce the ultimate physical address (the translation of the original guest-physical address); step 2 is executed.

2. Once the ultimate physical address is determined, the privileges determined by the EPT paging-structure entries are evaluated:

   a. If the access to the guest-physical address is not allowed by these privileges (see Section 24.2.4.2), an EPT violation occurs.

   b. If the access to the guest-physical address is allowed by these privileges, memory is accessed using the ultimate physical address.

If CR0.PG = 1, the translation of a linear address is also an iterative process, with the processor first accessing an entry in the guest paging structure referenced by the guest-physical address in CR3, then accessing an entry in another guest paging structure referenced by the guest-physical address in the first guest paging-structure entry, etc. Each guest-physical address is itself translated using EPT and may cause an EPT-induced VM exit. The following items detail how page faults and EPT-induced VM exits are recognized during this iterative process:

1. An attempt is made to access a guest paging-structure entry with a guest-physical address (initially, the address is derived from the one in CR3).

   a. If the access fails because of an EPT misconfiguration or an EPT violation (see above), an EPT-induced VM exit occurs.

   b. If the access does not cause an EPT-induced VM exit, bit 0 (the present flag) of the entry is consulted:

      i) If the present flag is 0 or any reserved bit is set, a page fault occurs.

      ii) If the present flag is 1, no reserved bit is set, operation depends on whether the entry references another guest paging structure (whether it is a guest PDE with PS = 1 or a guest PTE):

         - If the entry does references another guest paging structure, an entry from that structure is accessed; step 1 is executed for that other entry.
         - Otherwise, the entry is used to produce the ultimate guest-physical address (the translation of the original linear address); step 2 is executed.

2. Once the ultimate guest-physical address is determined, the privileges determined by the guest paging-structure entries are evaluated:

   a. If the access to the linear address is not allowed by these privileges (e.g., it was a write to a read-only page), a page fault occurs.

   b. If the access to the linear address is allowed by these privileges, an attempt is made to access memory at the ultimate guest-physical address:

      i) If the access fails because of an EPT misconfiguration or an EPT violation (see above), an EPT-induced VM exit occurs.

      ii) If the access does not cause an EPT-induced VM exit, memory is accessed using the ultimate physical address (the translation, using EPT, of the ultimate guest-physical address).

If CR0.PG = 0, a linear address is treated as a guest-physical address and is translated using EPT (see above). This process, if it completes without an EPT violation or EPT misconfiguration, produces a physical address and determines the privileges allowed by the EPT paging-structure entries. If these privileges do not allow the access to the physical address (see Section 24.2.4.2), an EPT violation occurs. Otherwise, memory is accessed using the physical address.

24.2.5.1 Memory Type Used for Accessing EPT Paging Structures

This section explains how the memory type is determined for accesses to the EPT paging structures. The determination is based first on the value of bit 30 (cache disable—CD) in control register CR0:

- If CR0.CD = 0, the memory type used for any such reference is the EPT paging-structure memory type, which is specified in bits 2:0 of the extended-page-table pointer (EPTP), a VM-execution control field (see Section 20.6.11). A value of 0 indicates the uncacheable type (UC), while a value of 6 indicates the write-back type (WB). Other values are reserved.
- If CR0.CD = 1, the memory type used for any such reference is uncacheable (UC).

The MTRRs have no effect on the memory type used for an access to an EPT paging structure.

24.2.5.2 Memory Type Used for Translated Guest-Physical Addresses

The **effective memory type** of a memory access using a guest-physical address (an access that is translated using EPT) is the memory type that is used to access memory. The effective memory type is based on the value of bit 30 (cache disable—CD) in control register CR0; the **last** EPT paging-structure entry used to translate the guest-physical address (either an EPT PDE with bit 7 set to 1 or an EPT PTE); and the PAT memory type (see below):

- The **PAT memory type** depends on the value of CR0.PG:
  - If CR0.PG = 0, the PAT memory type is WB (writeback).[1]
  - If CR0.PG = 1, the PAT memory type is the memory type selected from the IA32_PAT MSR as specified in Section 10.12.3, "Selecting a Memory Type from the PAT".[2]
- The **EPT memory type** is specified in bits 5:3 of the last EPT paging-structure entry: 0 = UC; 1 = WC; 4 = WT; 5 = WP; and 6 = WB. Other values are reserved and will cause an EPT misconfiguration (see Section 24.2.4).
- If CR0.CD = 0, the effective memory type depends upon the value of bit 6 of the last EPT paging-structure entry:

1. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, CR0.PG can be 0 in VMX non-root operation only if the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.
2. Table 10-11 in Section 10.12.3, "Selecting a Memory Type from the PAT" illustrates how the PAT memory type is selected based on the values of the PAT, PCD, and PWT bits in a page-table entry (or page-directory entry with PS = 1). For accesses to a guest paging-structure entry X, the PAT memory type is selected from the table by using a value of 0 for the PAT bit with the values of PCD and PWT from the paging-structure entry Y that references X (or from CR3 if X is in the root paging structure).

— If the value is 0, the effective memory type is the combination of the EPT memory type and the PAT memory type specified in Table 10-7 in Section 10.5.2.2, using the EPT memory type in place of the MTRR memory type.

— If the value is 1, the memory type used for the access is the EPT memory type. The PAT memory type is ignored.

- If CR0.CD = 1, the effective memory type is UC.

The MTRRs have no effect on the memory type used for an access to a guest-physical address.

24.3.2 Creating and Using Cached Translation Information

The following items detail the creation of the mappings described in the previous section:

- The following items describe the creation of mappings while EPT is not in use (including execution outside VMX non-root operation):
  — VPID-tagged mappings may be created. They are derived from the paging structures referenced (directly or indirectly) by the current value of CR3 and are associated with the current VPID.
  — No VPID-tagged mappings are created with information derived from paging-structure entries that are not present (bit 0 is 0) or that set reserved bits. For example, if a PTE is not present, no VPID-tagged mapping will be created for any linear page number whose translation would use that PTE.
  — No EPTP-tagged or dual-tagged mappings are created while EPT is not in use.
- The following items describe the creation of mappings while EPT is in use:
  — EPTP-tagged mappings may be created. They are derived from the EPT paging structures referenced (directly or indirectly) by the current EPTP and are associated with that EPTP.
  — Dual-tagged mappings may be created. They are derived from the EPT paging structures referenced (directly or indirectly) by the current EPTP. If CR0.PG = 1, they are also derived from the paging structures referenced (directly or indirectly) by the current value of CR3. They are associated with the current VPID and the current EPTP.[1] No dual-tagged paging-structure-cache entries are created if CR0.PG = 0.[2]
  — No EPTP-tagged mappings or dual-tagged mappings are created with information derived from EPT paging-structure entries that are not present (bits 2:0 are all 0) or that are misconfigured (see Section 24.2.4.1).
  — No dual-tagged mappings are created with information derived from guest paging-structure entries that are not present or that set reserved bits.
  — No VPID-tagged mappings are created while EPT is in use.

1. At any given time, a logical processor may cache dual-tagged mappings for a VPID that are associated with different EPTPs. Similarly, it may cache dual-tagged mappings for an EPTP that are associated with different VPIDs.
2. If the capability MSR IA32_VMX_CR0_FIXED1 reports that CR0.PG must be 1 in VMX operation, CR0.PG can be 0 in VMX non-root operation only if the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls are both 1.

The following items detail the use of the various mappings:

- If EPT is not in use (e.g., when outside VMX non-root operation), a logical processor may use cached mappings as follows:
  - For accesses using linear addresses, it may use VPID-tagged mappings associated with the current VPID.
  - No EPTP-tagged or dual-tagged mappings are used while EPT is not in use.
- If EPT is in use, a logical processor may use cached mappings as follows:
  - For accesses using linear addresses, it may use dual-tagged mappings associated with the current VPID and the current EPTP.
  - For accesses using guest-physical addresses, it may use EPTP-tagged mappings associated with the current EPTP.
  - No VPID-tagged mappings are used while EPT is in use.

24.3.3.2 Operations that Need Not Invalidate Cached Mappings

The following items detail cases of operations that are not required to invalidate certain cached mappings:

- Operations that architecturally invalidate entries in the TLBs or paging-structure caches independent of VMX operation are not required to invalidate any EPTP-tagged mappings.
- The INVVPID instruction is not required to invalidate any EPTP-tagged mappings.
- The INVEPT instruction is not required to invalidate any VPID-tagged mappings.
- VMX transitions are not required to invalidate any EPTP-tagged mappings. If the "enable VPID" VM-execution control is 1, VMX transitions are not required to invalidate any VPID-tagged mappings or dual-tagged mappings.
- The VMXOFF and VMXON instructions are not required to invalidate any VPID-tagged mappings, EPTP-tagged mappings, or dual-tagged mappings.

A logical processor may invalidate any cached mappings at any time. For this reason, the operations identified above may invalidate the indicated mappings despite the fact that doing so is not required.

17. Real-mode Virtualization Updates to Chapter 25 (Chapter 25, Vol 3B)

Change bars show changes to Chapter 25 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

25.1 SYSTEM MANAGEMENT MODE OVERVIEW

SMM is a special-purpose operating mode provided for handling system-wide functions like power management, system hardware control, or proprietary OEM-designed code. It is intended for use only by system firmware, not by applications software or general-purpose systems software. The main benefit of SMM is that it offers a distinct and easily isolated processor environment that operates transparently to the operating system or executive and software applications.

When SMM is invoked through a system management interrupt (SMI), the processor saves the current state of the processor (the processor's context), then switches to a separate operating environment contained in system management RAM (SMRAM). While in SMM, the processor executes SMI handler code to perform operations such as powering down unused disk drives or monitors, executing proprietary code, or placing the whole system in a suspended state. When the SMI handler has completed its operations, it executes a resume (RSM) instruction. This instruction causes the processor to reload the saved context of the processor, switch back to protected or real mode, and resume executing the interrupted application or operating-system program or task.

The following SMM mechanisms make it transparent to applications programs and operating systems:

- The only way to enter SMM is by means of an SMI.
- The processor executes SMM code in a separate address space (SMRAM) that can be made inaccessible from the other operating modes.
- Upon entering SMM, the processor saves the context of the interrupted program or task.
- All interrupts normally handled by the operating system are disabled upon entry into SMM.
- The RSM instruction can be executed only in SMM.

SMM is similar to real-address mode in that there are no privilege levels or address mapping. An SMM program can address up to 4 GBytes of memory and can execute all I/O and applicable system instructions. See Section 25.5 for more information about the SMM execution environment.

NOTES

Software developers should be aware that, even if a logical processor was using the physical-address extension (PAE) mechanism (introduced in the P6 family processors) or was in IA-32e mode before an SMI, this will not be the case after the SMI is delivered. This is because delivery of an SMI disables paging (see Table 25-4). (This does not apply if the dual-monitor treatment of SMIs and SMM is active; see Section 25.15.)

25.3.2 Exiting From SMM

The only way to exit SMM is to execute the RSM instruction. The RSM instruction is only available to the SMI handler; if the processor is not in SMM, attempts to execute the RSM instruction result in an invalid-opcode exception (#UD) being generated.

The RSM instruction restores the processor's context by loading the state save image from SMRAM back into the processor's registers. The processor then returns an SMIACK transaction on the system bus and returns program control back to the interrupted program.

Upon successful completion of the RSM instruction, the processor signals external hardware that SMM has been exited. For the P6 family processors, an SMI acknowledge transaction is generated on the system bus and the multiplexed status signal EXF4 is no longer generated on bus cycles. For the Pentium and Intel486 processors, the SMIACT# pin is deserted.

If the processor detects invalid state information saved in the SMRAM, it enters the shutdown state and generates a special bus cycle to indicate it has entered shutdown state. Shutdown happens only in the following situations:

- A reserved bit in control register CR4 is set to 1 on a write to CR4. This error should not happen unless SMI handler code modifies reserved areas of the SMRAM saved state map (see Section 25.4.1). CR4 is saved in the state map in a reserved location and cannot be read or modified in its saved state.
- An illegal combination of bits is written to control register CR0, in particular PG set to 1 and PE set to 0, or NW set to 1 and CD set to 0.
- (For the Pentium and Intel486 processors only.) If the address stored in the SMBASE register when an RSM instruction is executed is not aligned on a 32-KByte boundary. This restriction does not apply to the P6 family processors.

In the shutdown state, Intel processors stop executing instructions until a RESET#, INIT# or NMI# is asserted. While Pentium family processors recognize the SMI# signal in shutdown state, P6 family and Intel486 processors do not. Intel does not support using SMI# to recover from shutdown states for any processor family; the response of processors in this circumstance is not well defined. On Pentium 4 and later processors, shutdown will inhibit INTR and A20M but will not change any of the other inhibits. On these processors, NMIs will be inhibited if no action is taken in the SMM handler to uninhibit them (see Section 25.8).

If the processor is in the HALT state when the SMI is received, the processor handles the return from SMM slightly differently (see Section 25.8). Also, the SMBASE address can be changed on a return from SMM (see Section 25.11).

25.4.2 SMRAM Caching

An IA-32 processor does not automatically write back and invalidate its caches before entering SMM or before exiting SMM. Because of this behavior, care must be taken in the placement of the SMRAM in system memory and in the caching of the SMRAM to prevent cache incoherence when switching back and forth between SMM and protected mode operation. Either of the following three methods of locating the SMRAM in system memory will guarantee cache coherency:

- Place the SRAM in a dedicated section of system memory that the operating system and applications are prevented from accessing. Here, the SRAM can be designated as

cacheable (WB, WT, or WC) for optimum processor performance, without risking cache incoherence when entering or exiting SMM.

- Place the SRAM in a section of memory that overlaps an area used by the operating system (such as the video memory), but designate the SMRAM as uncacheable (UC). This method prevents cache access when in SMM to maintain cache coherency, but the use of uncacheable memory reduces the performance of SMM code.
- Place the SRAM in a section of system memory that overlaps an area used by the operating system and/or application code, but explicitly flush (write back and invalidate) the caches upon entering and exiting SMM mode. This method maintains cache coherency, but the incurs the overhead of two complete cache flushes.

For Pentium 4, Intel Xeon, and P6 family processors, a combination of the first two methods of locating the SMRAM is recommended. Here the SMRAM is split between an overlapping and a dedicated region of memory. Upon entering SMM, the SMRAM space that is accessed overlaps video memory (typically located in low memory). This SMRAM section is designated as UC memory. The initial SMM code then jumps to a second SMRAM section that is located in a dedicated region of system memory (typically in high memory). This SMRAM section can be cached for optimum processor performance.

For systems that explicitly flush the caches upon entering SMM (the third method described above), the cache flush can be accomplished by asserting the FLUSH# pin at the same time as the request to enter SMM (generally initiated by asserting the SMI# pin). The priorities of the FLUSH# and SMI# pins are such that the FLUSH# is serviced first. To guarantee this behavior, the processor requires that the following constraints on the interaction of FLUSH# and SMI# be met. In a system where the FLUSH# and SMI# pins are synchronous and the set up and hold times are met, then the FLUSH# and SMI# pins may be asserted in the same clock. In asynchronous systems, the FLUSH# pin must be asserted at least one clock before the SMI# pin to guarantee that the FLUSH# pin is serviced first.

Upon leaving SMM (for systems that explicitly flush the caches), the WBINVD instruction should be executed prior to leaving SMM to flush the caches.

NOTES

In systems based on the Pentium processor that use the FLUSH# pin to write back and invalidate cache contents before entering SMM, the processor will prefetch at least one cache line in between when the Flush Acknowledge cycle is run and the subsequent recognition of SMI# and the assertion of SMIACT#.

It is the obligation of the system to ensure that these lines are not cached by returning KEN# inactive to the Pentium processor.

25.4.2.1 System Management Range Registers (SMRR)

SMI handler code and data stored by SMM code resides in SMRAM. The SMRR interface is an enhancement in Intel 64 architecture to limit cacheable reference of addresses in SMRAM to code running in SMM. The SMRR interface can be configured only by code running in SMM. Details of SMRR is described in Section 10.11.2.4.

25.7.1 I/O State Implementation

Within the extended SMM state save map, a bit (IO_SMI) is provided that is set only when an SMI is either taken immediately after a *successful* I/O instruction or is taken after a *successful* iteration of a REP I/O instruction (the *successful* notion pertains to the processor point of view; not necessarily to the corresponding platform function). When set, the IO_SMI bit provides a strong indication that the corresponding SMI was synchronous. In this case, the SMM State Save Map also supplies the port address of the I/O operation. The IO_SMI bit and the I/O Port Address may be used in conjunction with the information logged by the platform to confirm that the SMI was indeed synchronous.

The IO_SMI bit by itself is a strong indication, not a guarantee, that the SMI is synchronous. This is because an asynchronous SMI might coincidentally be taken after an I/O instruction. In such a case, the IO_SMI bit would still be set in the SMM state save map.

Information characterizing the I/O instruction is saved in two locations in the SMM State Save Map (Table 25-5). The IO_SMI bit also serves as a valid bit for the rest of the I/O information fields. The contents of these I/O information fields are not defined when the IO_SMI bit is not set.

25.10 AUTO HALT RESTART

If the processor is in a HALT state (due to the prior execution of a HLT instruction) when it receives an SMI, the processor records the fact in the auto HALT restart flag in the saved processor state (see Figure 25-3). (This flag is located at offset 7F02H and bit 0 in the state save area of the SMRAM.)

If the processor sets the auto HALT restart flag upon entering SMM (indicating that the SMI occurred when the processor was in the HALT state), the SMI handler has two options:

- It can leave the auto HALT restart flag set, which instructs the RSM instruction to return program control to the HLT instruction. This option in effect causes the processor to re-enter the HALT state after handling the SMI. (This is the default operation.)
- It can clear the auto HALT restart flag, with instructs the RSM instruction to return program control to the instruction following the HLT instruction.

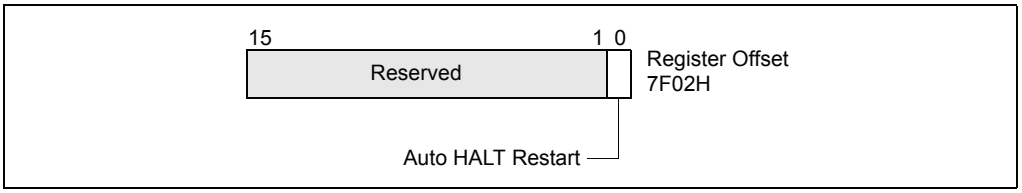

Figure 25-3 Auto HALT Restart Field

These options are summarized in Table 25-7. If the processor was not in a HALT state when the SMI was received (the auto HALT restart flag is cleared), setting the flag to 1 will cause unpredictable behavior when the RSM instruction is executed.

25.12 I/O INSTRUCTION RESTART

If the I/O instruction restart flag in the SMM revision identifier field is set (see Section 25.9), the I/O instruction restart mechanism is present on the processor. This mechanism allows an interrupted I/O instruction to be re-executed upon returning from SMM mode. For example, if an I/O instruction is used to access a powered-down I/O device, a chip set supporting this device can intercept the access and respond by asserting SMI#. This action invokes the SMI handler to power-up the device. Upon returning from the SMI handler, the I/O instruction restart mechanism can be used to re-execute the I/O instruction that caused the SMI.

The I/O instruction restart field (at offset 7F00H in the SMM state-save area, see Figure 25-5) controls I/O instruction restart. When an RSM instruction is executed, if this field contains the value FFH, then the EIP register is modified to point to the I/O instruction that received the SMI request. The processor will then automatically re-execute the I/O instruction that the SMI trapped. (The processor saves the necessary machine state to insure that re-execution of the instruction is handled coherently.)

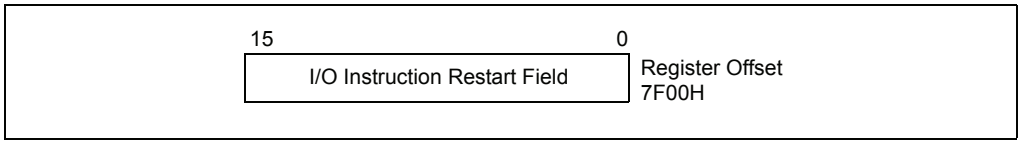

Figure 25-5 I/O Instruction Restart Field

If the I/O instruction restart field contains the value 00H when the RSM instruction is executed, then the processor begins program execution with the instruction following the I/O instruction. (When a repeat prefix is being used, the next instruction may be the next I/O instruction in the repeat loop.) Not re-executing the interrupted I/O instruction is the default behavior; the processor automatically initializes the I/O instruction restart field to 00H upon entering SMM. Table 25-8 summarizes the states of the I/O instruction restart field.

Table 25-8 I/O Instruction Restart Field Values

Value of Flag After Entry to SMM	Value of Flag When Exiting SMM	Action of Processor When Exiting SMM
00H	00H	Does not re-execute trapped I/O instruction.
00H	FFH	Re-executes trapped I/O instruction.

The I/O instruction restart mechanism does not indicate the cause of the SMI. It is the responsibility of the SMI handler to examine the state of the processor to determine the cause of the SMI and to determine if an I/O instruction was interrupted and should be restarted upon exiting SMM. If an SMI interrupt is signaled on a non-I/O instruction boundary, setting the I/O instruction restart field to FFH prior to executing the RSM instruction will likely result in a program error.

25.14.1 Default Treatment of SMI Delivery

Ordinary SMI delivery saves processor state into SMRAM and then loads state based on architectural definitions. Under the default treatment, processors that support VMX operation perform SMI delivery as follows:

```
enter SMM;
save the following internal to the processor:
    CR4.VMXE
    an indication of whether the logical processor was in VMX operation (root or non-root)
IF the logical processor is in VMX operation
    THEN
        save current VMCS pointer internal to the processor;
        leave VMX operation;
        save VMX-critical state defined below;
FI;
IF the logical processor supports SMX operation
    THEN
        save internal to the logical processor an indication of whether the Intel® TXT private space is
locked;
        IF the TXT private space is unlocked
            THEN lock the TXT private space;
        FI;
FI;
CR4.VMXE ← 0;
perform ordinary SMI delivery:
    save processor state in SMRAM;
    set processor state to standard SMM values;[1]
    invalidate VPID-tagged mappings and dual-tagged mappings associated with VPID 0000H; dual-
tagged mappings for VPID 0000H are invalidated for all EPTPs (see Section 24.3);
```

The pseudocode above makes reference to the saving of **VMX-critical state**. This state consists of the following: (1) SS.DPL (the current privilege level); (2) RFLAGS.VM[2]; (3) the state of blocking by STI and by MOV SS (see Table 20-3 in Section 20.4.2); (4) the state of virtual-NMI blocking (only if the processor is in VMX non-root operation and the "virtual NMIs" VM-execution control is 1); and (5) an indication of whether an MTF VM exit is pending (see Section 21.7.2). These data may be saved internal to the processor or in the VMCS region of the current VMCS. Processors that do not support SMI recognition while there is blocking by STI or by MOV SS need not save the state of such blocking.

If the logical processor supports the 1-setting of the "enable EPT" VM-execution control and the logical processor was in VMX non-root operation at the time of an SMI, it saves the value of that control into bit 0 of the 32-bit field at offset SMBASE + 8000H + 7EE0H (SMBASE + FEE0H; see Table 25-3).[3] If the logical processor was not in VMX non-root operation at the time of the SMI, it saves 0 into that bit. If the logical processor saves 1 into that bit (it was in VMX non-root operation and the "enable EPT" VM-execution

1. This causes the logical processor to block INIT signals, NMIs, and SMIs.
2. Section 25.14 and Section 25.15 use the notation RAX, RIP, RSP, RFLAGS, etc. for processor registers because most processors that support VMX operation also support Intel 64 architecture. For processors that do not support Intel 64 architecture, this notation refers to the 32-bit forms of these registers (EAX, EIP, ESP, EFLAGS, etc.). In a few places, notation such as EAX is used to refer specifically to the lower 32 bits of the register.

control was 1), it saves the value of the EPT pointer (EPTP) into the 64-bit field at offset SMBASE + 8000H + 7ED8H (SMBASE + FED8H).

Because SMI delivery causes a logical processor to leave VMX operation, all the controls associated with VMX non-root operation are disabled in SMM and thus cannot cause VM exits while the logical processor in SMM.

25.14.2 Default Treatment of RSM

Ordinary execution of RSM restores processor state from SMRAM. Under the default treatment, processors that support VMX operation perform RSM as follows:

```
IF VMXE = 1 in CR4 image in SMRAM
    THEN fail and enter shutdown state;
    ELSE
        restore state normally from SMRAM;
        invalidate VPID-tagged mappings and dual-tagged mappings associated with all VPIDs; dual-
tagged mappings are invalidated for all EPTPs (see Section 24.3);
        IF the logical processor supports SMX operation andthe Intel® TXT private space was unlocked
at the time of the last SMI (as saved)
            THEN unlock the TXT private space;
        FI;
        CR4.VMXE ← value stored internally;
        IF internal storage indicates that the logical processor
        had been in VMX operation (root or non-root)
            THEN
                enter VMX operation (root or non-root);
                restore VMX-critical state as defined in Section 25.14.1;
                set to their fixed values any bits in CR0 and CR4 whose values must be fixed in VMX
operation (see Section 19.8);[1]
                IF RFLAGS.VM = 0 AND (in VMX root operation OR the "unrestricted guest" VM-
execution control is 0)[2]
                    THEN
                        CS.RPL ← SS.DPL;
                        SS.RPL ← SS.DPL;
                FI;
                restore current VMCS pointer;
        FI;
        leave SMM;
        IF logical processor will be in VMX operation or in SMX operation after RSM
```

---

3. "Enable EPT" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, SMI functions as the "enable EPT" VM-execution control were 0. See Section 20.6.2.

1. If the RSM is to VMX non-root operation and both the "unrestricted guest" VM-execution control and bit 31 of the primary processor-based VM-execution controls will be 1, CR0.PE and CR0.PG retain the values that were loaded from SMRAM regardless of what is reported in the capability MSR IA32_VMX_CR0_FIXED1.

2. "Unrestricted guest" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VM entry functions as if the "unrestricted guest" VM-execution control were 0. See Section 20.6.2.

```
            THEN block A20M and leave A20M mode;
        FI;
FI;
```

RSM unblocks SMIs. It restores the state of blocking by NMI (see Table 20-3 in Section 20.4.2) as follows:

- If the RSM is not to VMX non-root operation or if the "virtual NMIs" VM-execution control will be 0, the state of NMI blocking is restored normally.
- If the RSM is to VMX non-root operation and the "virtual NMIs" VM-execution control will be 1, NMIs are not blocked after RSM. The state of virtual-NMI blocking is restored as part of VMX-critical state.

INIT signals are blocked after RSM if and only if the logical processor will be in VMX root operation.

If RSM returns a logical processor to VMX non-root operation, it re-establishes the controls associated with the current VMCS. If the "interrupt-window exiting" VM-execution control is 1, a VM exit occurs immediately after RSM if the enabling conditions apply. The same is true for the "NMI-window exiting" VM-execution control. Such VM exits occur with their normal priority. See Section 21.3.

If an MTF VM exit was pending at the time of the previous SMI, an MTF VM exit is pending on the instruction boundary following execution of RSM. The following items detail the treatment of MTF VM exits that may be pending following RSM:

- System-management interrupts (SMIs), INIT signals, and higher priority events take priority over these MTF VM exits. These MTF VM exits take priority over debug-trap exceptions and lower priority events.
- These MTF VM exits wake the logical processor if RSM caused the logical processor to enter the HLT state (see Section 25.10). They do not occur if the logical processor just entered the shutdown state.

25.15.2.5 Updating Non-Register State

SMM VM exits affect the non-register state of a logical processor as follows:

- SMM VM exits cause non-maskable interrupts (NMIs) to be blocked; they may be unblocked through execution of IRET or through a VM entry (depending on the value loaded for the interruptibility state and the setting of the "virtual NMIs" VM-execution control).
- SMM VM exits cause SMIs to be blocked; they may be unblocked by a VM entry that returns from SMM (see Section 25.15.4).

SMM VM exits invalidate VPID-tagged mappings and dual-tagged mappings associated with VPID 0000H (dual-tagged mappings for VPID 0000H are invalidated for all EPTPs); see Section 24.3. (Ordinary VM exits are not required to perform such invalidation if the "enable VPID" VM-execution control is 1; see Section 23.5.5.)

25.15.4.4 Checks on the Guest State Area

Section 22.3.1 specifies checks performed on fields in the guest-state area of the VMCS. Some of these checks are conditioned on the settings of certain VM-execution controls (e.g., "virtual NMIs" or "unrestricted guest"). VM entries that return from SMM modify these checks based on whether the executive-VMCS pointer field contains the VMXON pointer:[1]

- If the executive-VMCS pointer field contains the VMXON pointer (the VM entry remains in VMX root operation), the checks are performed as all relevant VM-execution controls were 0. (As a result, some checks may not be performed at all.)
- If the executive-VMCS pointer field does not contain the VMXON pointer (the VM entry enters VMX non-root operation), this check is performed based on the settings of the VM-execution controls in the executive VMCS (the VMCS referenced by the executive-VMCS pointer field in the current VMCS).

For VM entries that return from SMM, the activity-state field must not indicate the wait-for-SIPI state if the executive-VMCS pointer field contains the VMXON pointer (the VM entry is to VMX root operation).

25.14.4.5 Loading Guest State

VM entries that return from SMM load the SMBASE register from the SMBASE field.

VM entries that return from SMM invalidate VPID-tagged mappings and dual-tagged mappings associated with all VPIDs (dual-tagged mappings are invalidated for all EPTPs); see Section 24.3. (Ordinary VM entries are required to perform such invalidation only for VPID 0000H and are not required to do even that if the "enable VPID" VM-execution control is 1; see Section 22.3.2.5.)

25.15.5 Enabling the Dual-Monitor Treatment

Code and data for the SMM monitor reside in a region of SMRAM called the **monitor segment** (MSEG). Code running in SMM determines the location of MSEG and establishes its content. This code is also responsible for enabling the dual-monitor treatment.

SMM code enables the dual-monitor treatment and determines the location of MSEG by writing to IA32_SMM_MONITOR_CTL MSR (index 9BH). The MSR has the following format:

- Bit 0 is the register's valid bit. The SMM monitor may be invoked using VMCALL only if this bit is 1. Because VMCALL is used to activate the dual-monitor treatment (see Section 25.15.6), the dual-monitor treatment cannot be activated if the bit is 0. This bit is cleared when the logical processor is reset.
- Bits 11:1 are reserved.
- Bits 31:12 contain a value that, when shifted right 12 bits, is the physical address of MSEG (the MSEG base address).
- Bits 63:32 are reserved.

The following items detail use of this MSR:

1. An SMM monitor can determine the VMXON pointer by reading the executive-VMCS pointer field in the current VMCS after the SMM VM exit that activates the dual-monitor treatment.

- The IA32_SMM_MONITOR_CTL MSR is supported only on processors that support the dual-monitor treatment.[1] On other processors, accesses to the MSR using RDMSR or WRMSR generate a general-protection fault (#GP(0)).
- A write to the IA32_SMM_MONITOR_CTL MSR using WRMSR generates a general-protection fault (#GP(0)) if executed outside of SMM or if an attempt is made to set any reserved bit. An attempt to write to IA32_SMM_MONITOR_CTL MSR fails if made as part of a VM exit that does not end in SMM or part of a VM entry that does not begin in SMM.
- Reads from IA32_SMM_MONITOR_CTL MSR using RDMSR are allowed any time RDMSR is allowed. The MSR may be read as part of any VM exit.
- The dual-monitor treatment can be activated only if the valid bit in the MSR is set to 1.

The 32 bytes located at the MSEG base address are called the **MSEG header**. The format of the MSEG header is given in Table 25-10 (each field is 32 bits).

Table 25-10 Format of MSEG Header

Byte Offset	Field
0	MSEG-header revision identifier
4	SMM-monitor features
8	GDTR limit
12	GDTR base offset
16	CS selector
20	EIP offset
24	ESP offset
28	CR3 offset

To ensure proper behavior in VMX operation, software should maintain the MSEG header in writeback cacheable memory. Future implementations may allow or require a different memory type.[2] Software should consult the VMX capability MSR IA32_VMX_BASIC (see Appendix G.1).

SMM code should enable the dual-monitor treatment (by setting the valid bit in IA32_SMM_MONITOR_CTL MSR) only after establishing the content of the MSEG header as follows:

- Bytes 3:0 contain the **MSEG revision identifier**. Different processors may use different MSEG revision identifiers. These identifiers enable software to avoid using an MSEG header formatted for one processor on a processor that uses a different

1. Software should consult the VMX capability MSR IA32_VMX_BASIC (see Appendix G.1) to determine whether the dual-monitor treatment is supported.
2. Alternatively, software may map the MSEG header with the UC memory type; this may be necessary, depending on how memory is organized. Doing so is strongly discouraged unless necessary as it will cause the performance of transitions using those structures to suffer significantly. In addition, the processor will continue to use the memory type reported in the VMX capability MSR IA32_VMX_BASIC with exceptions noted in Appendix G.1.

format. Software can discover the MSEG revision identifier that a processor uses by reading the VMX capability MSR IA32_VMX_MISC (see Appendix G.6).

- Bytes 7:4 contain the **SMM-monitor features** field. Bits 31:1 of this field are reserved and must be zero. Bit 0 of the field is the **IA-32e mode SMM feature bit**. It indicates whether the logical processor will be in IA-32e mode after the SMM monitor is activated (see Section 25.15.6).
- Bytes 31:8 contain fields that determine how processor state is loaded when the SMM monitor is activated (see Section 25.15.6.4). SMM code should establish these fields so that activating of the SMM monitor invokes the SMM monitor's initialization code.

18. Intel® Microarchitecture core and uncore events additions to Appendix A (Appendix A, Vol 3B)

Change bars show changes to Appendix A of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

This appendix lists the performance-monitoring events that can be monitored with the Intel 64 or IA-32 processors. The ability to monitor performance events and the events that can be monitored in these processors are mostly model-specific, except for architectural performance events, described in Section A.1.

Non-architectural performance events (i.e. model-specific events) are listed for each generation of microarchitecture:

- Section A.2 - Processors based on Intel microarchitecture (Nehalem)
- Section A.3 - Processors based on Enhanced Intel Core microarchitecture
- Section A.4 - Processors based on Intel Core microarchitecture
- Section A.5 - Processors based on Intel Atom microarchitecture
- Section A.6 - Intel Core Solo and Intel Core Duo processors
- Section A.7 - Processors based on Intel NetBurst microarchitecture
- Section A.8 - Pentium M family processors
- Section A.9 - P6 family processors
- Section A.10 - Pentium processors

A.1 ARCHITECTURAL PERFORMANCE-MONITORING EVENTS

Architectural performance events are introduced in Intel Core Solo and Intel Core Duo processors. They are also supported on processors based on Intel Core microarchitecture. Table A-1 lists pre-defined architectural performance events that can be configured using general-purpose performance counters and associated event-select registers.

Table A-1 Architectural Performance Events

Event Num.	Event Mask Mnemonic	Umask Value	Description	Comment
3CH	UnHalted Core Cycles	00H	Unhalted core cycles	
3CH	UnHalted Reference Cycles	01H	Unhalted reference cycles	Measures bus cycle[1]
C0H	Instruction Retired	00H	Instruction retired	
2EH	LLC Reference	4FH	LL cache references	
2EH	LLC Misses	41H	LL cache misses	
C4H	Branch Instruction Retired	00H	Branch instruction retired	
C5H	Branch Misses Retired	00H	Mispredicted Branch Instruction retired	

NOTES:
1. Implementation of this event in Intel Core 2 processor family, Intel Core Duo, and Intel Core Solo processors measures bus clocks.

A.2 PERFORMANCE MONITORING EVENTS FOR INTEL® CORE™I7 PROCESSOR FAMILY

Processors based on the Intel microarchitecture (Nehalem) support the architectural and non-architectural performance-monitoring events listed in Table A-1 and Table A-2. In addition, they also support the following non-architectural, product-specific uncore performance-monitoring events listed in Table A-3. Fixed counters support the architecture events defined in Table A-5.

Table A-2 Non-Architectural Performance Events In the Processor Core for Intel Core i7 Processor and Intel Xeon Processor 5500 Series

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
0FH	08H	MEM_UNCORE_RETIRED.REMOTE_CACHE_LOCAL_HOME_HIT	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and L3 caches and HIT in a remote socket's cache. Only counts locally homed lines.	
0FH	10H	MEM_UNCORE_RETIRED.REMOTE_DRAM	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and L3 caches and was remotely homed. This includes both DRAM access and HITM in a remote socket's cache for remotely homed lines.	
0FH	20H	MEM_UNCORE_RETIRED.LOCAL_DRAM	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and L3 caches and required a local socket memory reference. This includes locally homed cachelines that were in a modified state in another socket.	
AEH	01H	ITLB_FLUSH	Counts the number of ITLB flushes	
F0H	40H	L2_TRANSACTIONS.WB	Counts L2 writeback operations to the L3.	

Non-architectural Performance monitoring events that are located in the uncore subsystem may be product implementation specific between different platforms using processors based on Intel microarchitecture (Nehalem). Lists performance events available to Intel Core i7 processors.

Table A-3 Non-Architectural Performance Events In the Processor Uncore for Intel Core i7 Processor and Intel Xeon Processor 5500 Series

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
0BH	10H	UNC_L3_LINES_OUT.F_STATE	Counts the number of L3 lines victimized that were in the F state.	
0BH	1FH	UNC_L3_LINES_OUT.ANY	Counts the number of L3 lines victimized in any state.	
20H	02H	UNC_QHL_REQUESTS.IOH_WRITES	Counts number of Quickpath Home Logic write requests from the IOH.	
20H	08H	UNC_QHL_REQUESTS.REMOTE_WRITES	Counts number of Quickpath Home Logic write requests from a remote socket.	
21H	01H	UNC_QHL_CYCLES_FULL.IOH	Counts uclk cycles all entries in the Quickpath Home Logic IOH are full.	
22H	01H	UNC_QHL_CYCLES_NOT_EMPTY.IOH	Counts uclk cycles all entries in the Quickpath Home Logic IOH is empty.	
22H	02H	UNC_QHL_CYCLES_NOT_EMPTY.REMOTE	Counts uclk cycles all entries in the Quickpath Home Logic remote tracker is empty.	
22H	04H	UNC_QHL_CYCLES_NOT_EMPTY.LOCAL	Counts uclk cycles all entries in the Quickpath Home Logic local tracker is empty.	
23H	01H	UNC_QHL_OCCUPANCY.IOH	QHL IOH tracker allocate to deallocate read occupancy.	
23H	02H	UNC_QHL_OCCUPANCY.REMOTE	QHL remote tracker allocate to deallocate read occupancy.	
23H	04H	UNC_QHL_OCCUPANCY.LOCAL	QHL local tracker allocate to deallocate read occupancy.	
2AH	01H	UNC_QMC_OCCUPANCY.CH0	IMC channel 0 normal read request occupancy.	
2AH	02H	UNC_QMC_OCCUPANCY.CH1	IMC channel 1 normal read request occupancy.	
2AH	04H	UNC_QMC_OCCUPANCY.CH2	IMC channel 2 normal read request occupancy.	
2BH	01H	UNC_QMC_ISSOC_OCCUPANCY.CH0	IMC channel 0 issoc read request occupancy.	
2BH	02H	UNC_QMC_ISSOC_OCCUPANCY.CH1	IMC channel 1 issoc read request occupancy.	
2BH	04H	UNC_QMC_ISSOC_OCCUPANCY.CH2	IMC channel 2 issoc read request occupancy.	
2BH	07H	UNC_QMC_ISSOC_READS.ANY	IMC issoc read request occupancy.	

A.3 PERFORMANCE MONITORING EVENTS FOR INTEL® XEON® PROCESSOR 5200, 5400 SERIES AND INTEL® CORE™2 EXTREME PROCESSORS QX 9000 SERIES

Processors based on the Enhanced Intel Core microarchitecture support the architectural and non-architectural performance-monitoring events listed in Table A-1 and Table A-6. In addition, they also support the following non-architectural performance-monitoring events listed in Table A-4. Fixed counters support the architecture events defined in Table A-5.

Table A-5 Non-Architectural Performance Events for Processors based on Enhanced Intel Core Microarchitecture

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
C0H	08H	INST_RETIRED.VM_HOST	Instruction retired while in VMX root operations	
D2H	10H	RAT_STAALS.OTHER_SERIALIZATION_STALLS	This events counts the number of stalls due to other RAT resource serialization not counted by Umask value 0FH.	

A.4 PERFORMANCE MONITORING EVENTS FOR INTEL® XEON® PROCESSOR 3000, 3200, 5100, 5300 SERIES AND INTEL® CORE™2 DUO PROCESSORS

Processors based on the Intel Core microarchitecture support architectural and non-architectural performance-monitoring events.

Fixed-function performance counters are introduced first on processors based on Intel Core microarchitecture. Table A-5 lists pre-defined performance events that can be counted using fixed-function performance counters.

Table A-5 Fixed-Function Performance Counter and Pre-defined Performance Events

Fixed-Function Performance Counter	Address	Event Mask Mnemonic	Description
MSR_PERF_FIXED_CTR0	309H	Instr_Retired.Any	This event counts the number of instructions that retire execution. For instructions that consist of multiple micro-ops, this event counts the retirement of the last micro-op of the instruction. The counter continue counting during hardware interrupts, traps, and inside interrupt handlers

Table A-5 Fixed-Function Performance Counter and Pre-defined Performance Events (Continued)

Fixed-Function Performance Counter	Address	Event Mask Mnemonic	Description
MSR_PERF_FIXED_CTR1	30AH	CPU_CLK_UNHALTED.CORE	This event counts the number of core cycles while the core is not in a halt state. The core enters the halt state when it is running the HLT instruction. This event is a component in many key event ratios. The core frequency may change from time to time due to transitions associated with Enhanced Intel SpeedStep Technology or TM2. For this reason this event may have a changing ratio with regards to time. When the core frequency is constant, this event can approximate elapsed time while the core was not in halt state.
MSR_PERF_FIXED_CTR2	30BH	CPU_CLK_UNHALTED.REF	This event counts the number of reference cycles when the core is not in a halt state and not in a TM stop-clock state. The core enters the halt state when it is running the HLT instruction or the MWAIT instruction.
			This event is not affected by core frequency changes (e.g., P states) but counts at the same frequency as the time stamp counter. This event can approximate elapsed time while the core was not in halt state and not in a TM stop-clock state. This event has a constant ratio with the CPU_CLK_UNHALTED.BUS event.

Table A-6 lists general-purpose non-architectural performance-monitoring events supported in processors based on Intel Core microarchitecture. For convenience, Table A-6 also includes architectural events and describes minor model-specific behavior where applicable. Software must use a general-purpose performance counter to count events listed in Table A-6.

A.6 PERFORMANCE MONITORING EVENTS FOR INTEL® CORE™ SOLO AND INTEL® CORE™ DUO PROCESSORS

Table A-8 lists non-architectural performance events for Intel Core Duo processors. If a non-architectural event requires qualification in core specificity, it is indicated in the comment column. Table A-8 also applies to Intel Core Solo processors; bits in the unit mask corresponding to core-specificity are reserved and should be 00B.

Table A-8 Non-Architectural Performance Events in Intel Core Solo and Intel Core Duo Processors

Event Num.	Event Mask Mnemonic	Umask Value	Description	Comment
64H	Bus_Data_Rcv	40H	Number of data chunks received by this processor	
77H	Bus_Snoops	00H	Counts any snoop on the bus	Requires MESI qualification Requires agent specificity

19. Architectural PerfMon Updates and MSR Updates to Appendix B (Appendix B, Vol 3B)

Change bars show changes to Appendix B of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

Model specific registers and its bit-fields may be supported for a finite range of processor families/models. To distinguish between different processor family and/or models, software must use CPUID.01H leaf function to query the combination of "display family" and "display model" to determine model-specific availability of MSRs. The "display family" and "display model" are defined in the instruction reference pages of CPUID. Table B-1 lists the signature values of "display family" and "display model" for various processor families or processor number series.

Table B-1 CPUID Signature Values of DisplayFamily_DisplayModel

DisplayFamily_DisplayModel	Processor Families/Processor Number Series
06_1AH	Intel Core i7 Processor, Intel Xeon Processor 5500 series

Certain bit field position may be related to the maximum physical address width, the value of which is expressed as "MAXPHYWID" in Table B-2. "MAXPHYWID" is reported by CPUID.8000_0008H leaf.

Table B-2 IA-32 Architectural MSRs

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
E7H	231	IA32_MPERF	Maximum Qualified Performance Clock Counter (R/Write to clear)	If CPUID.06H: ECX[0] = 1
		63:0	**C0_MCNT: C0 Maximum Frequency Clock Count.** Increments at fixed interval (relative to TSC freq.) when the logical processor is in C0. Cleared upon overflow / wrap-around of IA32_APERF.	
E8H	232	IA32_APERF	Actual Performance Clock Counter (R/Write to clear)	If CPUID.06H: ECX[0] = 1
		63:0	**C0_ACNT: C0 Actual Frequency Clock Count.** Accumulates core clock counts at the coordinated clock frequency, when the logical processor is in C0. Cleared upon overflow / wrap-around of IA32_MPERF.	
FEH	254	IA32_MTRRCAP (MTRRcap)	MTRR Capability (RO) Section 10.11.2.1, "IA32_MTRR_DEF_TYPE MSR."	06_01H
		7:0	VCNT: The number of variable memory type ranges in the processor	
		8	Fixed range MTRRs are supported when set.	
		9	Reserved	
		10	WC Supported when set	
		11	SMRR Supported when set	
		63:12	Reserved	
174H	372	IA32_SYSENTER_CS	SYSENTER_CS_MSR (R/W)	06_01H
186H	390	IA32_PERFEVTSEL0 (PERFEVTSEL0)	Performance Event Select Register 0 (R/W)	If CPUID.0AH: EAX[15:8] > 0
		7:0	Event Select: Selects a performance event logic unit	

Table B-2 IA-32 Architectural MSRs (Continued)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		15:8	UMask: Qualifies the microarchitectural condition to detect on the selected event logic.	
		16	USR: Counts while in privilege level is not ring 0.	
		17	OS: Counts while in privilege level is ring 0.	
		18	Edge: Enables edge detection if set	
		19	PC: enables pin control	
		20	INT: enables interrupt on counter overflow	
		21	AnyThread: When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR.	
187H	391	IA32_PERFEVTSEL1 (PERFEVTSEL1)	Performance Event Select Register 1 (R/W)	If CPUID.0AH: EAX[15:8] > 1
1D9H	473	IA32_DEBUGCTL (MSR_DEBUGCTLA, MSR_DEBUGCTLB)	Trace/Profile Resource Control (R/W)	06_0EH
		0	LBR: Setting this bit to 1 enables the processor to record a running trace of the most recent branches taken by the processor in the LBR stack.	06_01H
		1	BTF: Setting this bit to 1 enables the processor to treat EFLAGS.TF as single-step on branches instead of single-step on instructions.	06_01H
		5:2	Reserved	
		6	TR: Setting this bit to 1 enables branch trace messages to be sent.	06_0EH

Table B-2 IA-32 Architectural MSRs (Continued)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		7	BTS: Setting this bit enables branch trace messages (BTMs) to be logged in a BTS buffer.	06_0EH
		8	BTINT: When clear, BTMs are logged in a BTS buffer in circular fashion. When this bit is set, an interrupt is generated by the BTS facility when the BTS buffer is full.	06_0EH
		9	1: BTS_OFF_OS: When set, BTS or BTM is skipped if CPL = 0.	06_0FH
		10	BTS_OFF_USR: When set, BTS or BTM is skipped if CPL > 0.	06_0FH
		11	FREEZE_LBRS_ON_PMI: When set, the LBR stack is frozen on a PMI request.	If CPUID.01H: ECX[15] = 1
		12	FREEZE_PERFMON_ON_PMI: When set, each ENABLE bit of the global counter control MSR are frozen (address 3BFH) on a PMI request	If CPUID.01H: ECX[15] = 1
		13	ENABLE_UNCORE_PMI: When set, enables the logical processor to receive and generate PMI on behalf of the uncore.	06_1AH
		14	FREEZE_WHILE_SMM: When set, freezes perfmon and trace messages while in SMM.	if IA32_PERF_CAPABILITIES[12] = '1
		63:15	Reserved	
1F2H	498	IA32_SMRR_PHYSBASE	**SMRR Base Address. (WO in SMM)** Base address of SMM memory range.	
		7:0	Type. Specifies memory type of the range.	
		11:8	Reserved.	

Table B-2 IA-32 Architectural MSRs (Continued)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		31:12	**PhysBase.** SMRR physical Base Address.	
		63:24	Reserved.	
1F3H	499	IA32_SMRR_PHYSMASK	**SMRR Range Mask. (WO in SMM)** Range Mask of SMM memory range.	
		10:0	Reserved.	
		11	**Valid.** Enable range mask	
		31:12	**PhysMask.** SMRR address range mask.	
		63:24	Reserved.	
1F8H	504	IA32_PLATFORM_DCA_CAP	DCA Capability (R)	06_0FH
345H	837	IA32_PERF_CAPABILITIES	RO	If CPUID.01H: ECX[15] = 1
		5:0	LBR format	
		6	PEBS Trap	
		7	PEBSSaveArchRegs	
4000_0000H - 4000_00FFH		Reserved MSR Address Space	**All existing and future processors will not implement MSR in this range**	

Table B-3 MSRs in Processors Based on Intel Core Microarchitecture

Register Address		Register Name	Shared/ Unique	Bit Description
Hex	Dec			
3AH	58	IA32_FEATURE_ CONTROL	Unique	**Control Features in Intel 64Processor. (R/ W).** see Table B-2
		3	Unique	**SMRR Enable. (R/WL).** When this bit is set and the lock bit is set makes the SMRR_PHYS_BASE and SMRR_PHYS_MASK registers read visible and writeable while in SMM.
A0H	160	MSR_SMRR_PHYS BASE	Unique	**System Management Mode Base Address register. (WO in SMM)** Model-specific implementation of SMRR-like interface, read visible and write only in SMM..
		11:0		Reserved
		31:12		PhysBase. SMRR physical Base Address.
		63:32		Reserved
A1H	161	MSR_SMRR_PHYS MASK	Unique	**System Management Mode Physical Address Mask register. (WO in SMM)** Model-specific implementation of SMRR-like interface, read visible and write only in SMM..
		10:0		Reserved
		11		Valid. Physical address base and range mask are valid.
		31:12		PhysMask. SMRR physical address range mask.
		63:32		Reserved
FEH	254	IA32_MTRRCAP	Unique	see Table B-2
		11	Unique	**SMRR Capability Using MSR 0A0H and 0A1H. (R)**

Table B-5 MSRs in Processors Based on Intel Microarchitecture (Nehalem)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
1A2H	418	MSR_ TEMPERATURE_TARGET	Unique	see Table B-2
		15:0		Reserved.
		23:16		**Temperature Target. (R)** The minimum temperature at which PROCHOT# will be asserted. The value is degree C.
		63:24		Reserved
1F2H	498	IA32_SMRR_PHYSBASE	Core	see Table B-2
1F3H	499	IA32_SMRR_PHYSMASK	Core	see Table B-2
301H	769	MSR_GQ_SNOOP_MESF	Package	
		0		**From M to S (R/W).**
		1		**From E to S (R/W).**
		2		**From S to S (R/W).**
		3		**From F to S (R/W).**
		4		**From M to I (R/W).**
		5		**From E to I (R/W).**
		6		**From S to I (R/W).**
		7		**From F to I (R/W).**
		63:8		Reserved

20. Updated MC0 decoding definitions for processors, CMCI table correction to Appendix E (Appendix E, Vol 3B)

Change bars show changes to Appendix G of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

Table E-2 Incremental Bus Error Codes of Machine Check for Processors Based on Intel Core Microarchitecture

Type	Bit No.	Bit Function	Bit Description
MCA error codes[1]	0-15		
Model specific errors	16-18	Reserved	Reserved
Model specific errors	19-24	Bus queue request type	'000001 for BQ_PREF_READ_TYPE error 000000 for BQ_DCU_READ_TYPE error 000010 for BQ_IFU_DEMAND_TYPE error 000011 for BQ_IFU_DEMAND_NC_TYPE error 000100 for BQ_DCU_RFO_TYPE error 000101 for BQ_DCU_RFO_LOCK_TYPE error 000110 for BQ_DCU_ITOM_TYPE error 001000 for BQ_DCU_WB_TYPE error 001010 for BQ_DCU_WCEVICT_TYPE error 001011 for BQ_DCU_WCLINE_TYPE error 001100 for BQ_DCU_BTM_TYPE error
			001101 for BQ_DCU_INTACK_TYPE error 001110 for BQ_DCU_INVALL2_TYPE error 001111 for BQ_DCU_FLUSHL2_TYPE error 010000 for BQ_DCU_PART_RD_TYPE error 010010 for BQ_DCU_PART_WR_TYPE error 010100 for BQ_DCU_SPEC_CYC_TYPE error 011000 for BQ_DCU_IO_RD_TYPE error 011001 for BQ_DCU_IO_WR_TYPE error 011100 for BQ_DCU_LOCK_RD_TYPE error 011110 for BQ_DCU_SPLOCK_RD_TYPE error 011101 for BQ_DCU_LOCK_WR_TYPE error 100100 for BQ_L2_WI_RFO_TYPE error 100110 for BQ_L2_WI_ITOM_TYPE error

Table E-2 Incremental Bus Error Codes of Machine Check for Processors Based on Intel Core Microarchitecture

Type	Bit No.	Bit Function	Bit Description
Model specific errors	27-25	Bus queue error type	'001 for Address Parity Error '010 for Response Hard Error '011 for Response Parity Error
Model specific errors	28	MCE Driven	1 if MCE is driven
	29	MCE Observed	1 if MCE is observed
	30	Internal BINIT	1 if BINIT driven for this processor
	31	BINIT Observed	1 if BINIT is observed for this processor
Other information	32-33	Reserved	Reserved
	34	PIC and FSB data parity	Data Parity detected on either PIC or FSB access
	35	Reserved	Reserved
	36	Response parity error	This bit is asserted in IA32_MC*i*_STATUS if this component has received a parity error on the RS[2:0]# pins for a response transaction. The RS signals are checked by the RSP# external pin.
	37	FSB address parity	Address parity error detected: 1 = Address parity error detected 0 = No address parity error
	38	Timeout BINIT	This bit is asserted in IA32_MC*i*_STATUS if this component has experienced a ROB time-out, which indicates that no micro-instruction has been retired for a predetermined period of time. A ROB time-out occurs when the 15-bit ROB time-out counter carries a 1 out of its high order bit. The timer is cleared when a micro-instruction retires, an exception is detected by the core processor, RESET is asserted, or when a ROB BINIT occurs.
			The ROB time-out counter is prescaled by the 8-bit PIC timer which is a divide by 128 of the bus clock the bus clock is 1:2, 1:3, 1:4 of the core clock). When a carry out of the 8-bit PIC timer occurs, the ROB counter counts up by one. While this bit is asserted, it cannot be overwritten by another error.
	39-41	Reserved	Reserved

Table E-2 Incremental Bus Error Codes of Machine Check for Processors Based on Intel Core Microarchitecture

Type	Bit No.	Bit Function	Bit Description
	42	Hard error	This bit is asserted in IA32_MC*i*_STATUS if this component has initiated a bus transactions which has received a hard error response. While this bit is asserted, it cannot be overwritten.
	43	IERR	This bit is asserted in IA32_MC*i*_STATUS if this component has experienced a failure that causes the IERR pin to be asserted. While this bit is asserted, it cannot be overwritten.
	44	Reserved	Reserved
	45	Reserved	Reserved
	46	Reserved	Reserved
	47-54	Reserved	Reserved
	55-56	Reserved	Reserved.
Status register validity indicators[1]	57-63		

NOTES:
1. These fields are architecturally defined. Refer to Chapter 14, "Machine-Check Architecture," for more information.

21. Real-mode Virtualization Updates to Appendix G (Appendix G, Vol 3B)

Change bars show changes to Appendix G of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*.

G.1 BASIC VMX INFORMATION

The IA32_VMX_BASIC MSR (index 480H) consists of the following fields:

- Bits 31:0 contain the 32-bit VMCS revision identifier used by the processor. Logical processors that use the same VMCS revision identifier use the same size for VMCS regions (see next item)
- Bits 44:32 report the number of bytes that software should allocate for the VMXON region and any VMCS region. It is a value greater than 0 and at most 4096 (bit 44 is set if and only if bits 43:32 are clear).
- Bit 48 indicates the width of the physical addresses that may be used for the VMXON region, each VMCS, and data structures referenced by pointers in a VMCS (I/O bitmaps, virtual-APIC page, MSR areas for VMX transitions). If the bit is 0, these addresses are limited to the processor's physical-address width.[1] If the bit is 1, these addresses are limited to 32 bits. This bit is always 0 for processors that support Intel 64 architecture and is always 1 for processors that do not support Intel 64 architecture.
- If bit 49 is read as 1, the logical processor supports the dual-monitor treatment of system-management interrupts and system-management mode. See Section 25.15 for details of this treatment.
- Bits 53:50 report the memory type that the logical processor uses to access the VMCS for VMREAD and VMWRITE and to access the VMCS, data structures referenced by pointers in the VMCS (I/O bitmaps, virtual-APIC page, MSR areas for VMX transitions), and the MSEG header during VM entries, VM exits, and in VMX non-root operation.[2]

  The first processors to support VMX operation use the write-back type. The values used are given in Table G-1.

Table G-1 Memory Types Used For VMCS Access

Value(s)	Field
0	Uncacheable (UC)
1–5	Not used
6	Write Back (WB)
7–15	Not used

1. On processors that support Intel 64 architecture, the pointer must not set bits beyond the processor's physical address width.
2. If the MTRRs are disabled by clearing the E bit (bit 11) in the IA32_MTRR_DEF_TYPE MSR, the logical processor uses the UC memory type to access the indicated data structures, regardless of the value reported in bits 53:50 in the IA32_VMX_BASIC MSR. The processor will also use the UC memory type if the setting of CR0.CD on this logical processor (or another logical processor on the same physical processor) would cause it to do so for all memory accesses. The values of IA32_MTRR_DEF_TYPE.E and CR0.CD do not affect the value reported in IA32_VMX_BASIC[53:50].

If software needs to access these data structures (e.g., to modify the contents of the MSR bitmaps), it can configure the paging structures to map them into the linear-address space. If it does so, it should establish mappings that use the memory type reported in this MSR.[1]

- If bit 54 is read as 1, the logical processor reports information in the VM-exit instruction-information field on VM exits due to execution of the INS and OUTS instructions. This reporting is done only if this bit is read as 1.
- Bit 55 is read as 1 if any VMX controls that default to 1 may be cleared to 0. See Appendix G.2 for details. It also reports support for the VMX capability MSRs IA32_VMX_TRUE_PINBASED_CTLS, IA32_VMX_TRUE_PROCBASED_CTLS, IA32_VMX_TRUE_EXIT_CTLS, and IA32_VMX_TRUE_ENTRY_CTLS. See Appendix G.3.1, Appendix G.3.2, Appendix G.4, and Appendix G.5 for details.
- The values of bits 47:45 and bits 63:56 are reserved and are read as 0.

G.3.1 Pin-Based VM-Execution Controls

The IA32_VMX_PINBASED_CTLS MSR (index 481H) reports on the allowed settings of most of the pin-based VM-execution controls (see Section 20.6.1):

- Bits 31:0 indicate the **allowed 0-settings** of these controls. VM entry fails if bit X is 0 in the pin-based VM-execution controls and bit X is 1 in this MSR.

  Exceptions are made for the pin-based VM-execution controls in the default1 class (see Appendix G.2). These are bits 1, 2, and 4; the corresponding bits of the IA32_VMX_PINBASED_CTLS MSR are always read as 1. The treatment of these controls by VM entry is determined by bit 55 in the IA32_VMX_BASIC MSR:

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 0, VM entry fails if any pin-based VM-execution control in the default1 class is 0.

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_PINBASED_CTLS MSR (see below) reports which of the pin-based VM-execution controls in the default1 class can be 0 on VM entry.
- Bits 63:32 indicate the **allowed 1-settings** of these controls. VM entry fails if bit X is 1 in the pin-based VM-execution controls and bit 32+X is 0 in this MSR.

If bit 55 in the IA32_VMX_BASIC MSR is read as 1,
the IA32_VMX_TRUE_PINBASED_CTLS MSR (index 48DH) reports on the allowed settings of all of the pin-based VM-execution controls:

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the pin-based VM-execution controls and bit X is 1 in this MSR. There are no exceptions.
- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the pin-based VM-execution controls and bit 32+X is 0 in this MSR.

It is necessary for software to consult only one of the capability MSRs to determine the allowed settings of the pin-based VM-execution controls:

1. Alternatively, software may map any of these regions or structures with the UC memory type. (This may be necessary for the MSEG header.) Doing so is discouraged unless necessary as it will cause the performance of software accesses to those structures to suffer. The processor will continue to use the memory type reported in the VMX capability MSR IA32_VMX_BASIC with the exceptions noted.

- If bit 55 in the IA32_VMX_BASIC MSR is read as 0, all information about the allowed settings of the pin-based VM-execution controls is contained in the IA32_VMX_PINBASED_CTLS MSR. (The IA32_VMX_TRUE_PINBASED_CTLS MSR is not supported.)
- If bit 55 in the IA32_VMX_BASIC MSR is read as 1, all information about the allowed settings of the pin-based VM-execution controls is contained in the IA32_VMX_TRUE_PINBASED_CTLS MSR. Assuming that software knows that the default1 class of pin-based VM-execution controls contains bits 1, 2, and 4, there is no need for software to consult the IA32_VMX_PINBASED_CTLS MSR.

G.3.2 Primary Processor-Based VM-Execution Controls

The IA32_VMX_PROCBASED_CTLS MSR (index 482H) reports on the allowed settings of most of the primary processor-based VM-execution controls (see Section 20.6.2):

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the primary processor-based VM-execution controls and bit X is 1 in this MSR.

  Exceptions are made for the primary processor-based VM-execution controls in the default1 class (see Appendix G.2). These are bits 1, 4–6, 8, 13–16, and 26; the corresponding bits of the IA32_VMX_PROCBASED_CTLS MSR are always read as 1. The treatment of these controls by VM entry is determined by bit 55 in the IA32_VMX_BASIC MSR:

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 0, VM entry fails if any of the primary processor-based VM-execution controls in the default1 class is 0.

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_PROCBASED_CTLS MSR (see below) reports which of the primary processor-based VM-execution controls in the default1 class can be 0 on VM entry.
- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the primary processor-based VM-execution controls and bit 32+X is 0 in this MSR.

If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_PROCBASED_CTLS MSR (index 48EH) reports on the allowed settings of all of the primary processor-based VM-execution controls:

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the primary processor-based VM-execution controls and bit X is 1 in this MSR. There are no exceptions.
- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the primary processor-based VM-execution controls and bit 32+X is 0 in this MSR.

It is necessary for software to consult only one of the capability MSRs to determine the allowed settings of the primary processor-based VM-execution controls:

- If bit 55 in the IA32_VMX_BASIC MSR is read as 0, all information about the allowed settings of the primary processor-based VM-execution controls is contained in the IA32_VMX_PROCBASED_CTLS MSR. (The IA32_VMX_TRUE_PROCBASED_CTLS MSR is not supported.)
- If bit 55 in the IA32_VMX_BASIC MSR is read as 1, all information about the allowed settings of the pin-based VM-execution controls is contained in the

IA32_VMX_TRUE_PROCBASED_CTLS MSR. Assuming that software knows that the default1 class of pin-based VM-execution controls contains bits 1, 4–6, 8, 13–16, and 26, there is no need for software to consult the IA32_VMX_PROCBASED_CTLS MSR.

G.4 VM-EXIT CONTROLS

The IA32_VMX_EXIT_CTLS MSR (index 483H) reports on the allowed settings of most of the VM-exit controls (see Section 20.7.1):

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the VM-exit controls and bit X is 1 in this MSR.

  Exceptions are made for the VM-exit controls in the default1 class (see Appendix G.2). These are bits 0–8, 10, 11, 13, 14, 16, and 17; the corresponding bits of the IA32_VMX_EXIT_CTLS MSR are always read as 1. The treatment of these controls by VM entry is determined by bit 55 in the IA32_VMX_BASIC MSR:

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 0, VM entry fails if any VM-exit control in the default1 class is 0.

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_EXIT_CTLS MSR (see below) reports which of the VM-exit controls in the default1 class can be 0 on VM entry.

- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the VM-exit controls and bit 32+X is 0 in this MSR.

If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_EXIT_CTLS MSR (index 48FH) reports on the allowed settings of all of the VM-exit controls:

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the VM-exit controls and bit X is 1 in this MSR. There are no exceptions.
- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the VM-exit controls and bit 32+X is 0 in this MSR.

It is necessary for software to consult only one of the capability MSRs to determine the allowed settings of the VM-exit controls:

- If bit 55 in the IA32_VMX_BASIC MSR is read as 0, all information about the allowed settings of the VM-exit controls is contained in the IA32_VMX_EXIT_CTLS MSR. (The IA32_VMX_TRUE_EXIT_CTLS MSR is not supported.)
- If bit 55 in the IA32_VMX_BASIC MSR is read as 1, all information about the allowed settings of the VM-exit controls is contained in the IA32_VMX_TRUE_EXIT_CTLS MSR. Assuming that software knows that the default1 class of VM-exit controls contains bits 0–8, 10, 11, 13, 14, 16, and 17, there is no need for software to consult the IA32_VMX_EXIT_CTLS MSR.

G.5 VM-ENTRY CONTROLS

The IA32_VMX_ENTRY_CTLS MSR (index 484H) reports on the allowed settings of most of the VM-entry controls (see Section 20.8.1):

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the VM-entry controls and bit X is 1 in this MSR.

  Exceptions are made for the VM-entry controls in the default1 class (see Appendix G.2). These are bits 0–8 and 12; the corresponding bits of the IA32_VMX_ENTRY_CTLS MSR are always read as 1. The treatment of these controls by VM entry is determined by bit 55 in the IA32_VMX_BASIC MSR:

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 0, VM entry fails if any VM-entry control in the default1 class is 0.

  — If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_ENTRY_CTLS MSR (see below) reports which of the VM-entry controls in the default1 class can be 0 on VM entry.
- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the VM-entry controls and bit 32+X is 0 in this MSR.

If bit 55 in the IA32_VMX_BASIC MSR is read as 1, the IA32_VMX_TRUE_ENTRY_CTLS MSR (index 490H) reports on the allowed settings of all of the VM-entry controls:

- Bits 31:0 indicate the allowed 0-settings of these controls. VM entry fails if bit X is 0 in the VM-entry controls and bit X is 1 in this MSR. There are no exceptions.
- Bits 63:32 indicate the allowed 1-settings of these controls. VM entry fails if bit X is 1 in the VM-entry controls and bit 32+X is 0 in this MSR.

It is necessary for software to consult only one of the capability MSRs to determine the allowed settings of the VM-entry controls:

- If bit 55 in the IA32_VMX_BASIC MSR is read as 0, all information about the allowed settings of the VM-entry controls is contained in the IA32_VMX_ENTRY_CTLS MSR. (The IA32_VMX_TRUE_ENTRY_CTLS MSR is not supported.)
- If bit 55 in the IA32_VMX_BASIC MSR is read as 1, all information about the allowed settings of the VM-entry controls is contained in the IA32_VMX_TRUE_ENTRY_CTLS MSR. Assuming that software knows that the default1 class of VM-entry controls contains bits 0–8 and 12, there is no need for software to consult the IA32_VMX_ENTRY_CTLS MSR.

G.6 MISCELLANEOUS DATA

The IA32_VMX_MISC MSR (index 485H) consists of the following fields:

- Bits 4:0 report a value X that specifies the relationship between the rate of the VMX-preemption timer and that of the timestamp counter (TSC). Specifically, the VMX-preemption timer (if it is active) counts down by 1 every time bit X in the TSC changes due to a TSC increment.
- If bit 5 is read as 1, VM exits store the value of IA32_EFER.LMA into the "IA-32e mode guest" VM-entry control; see Section 23.2 for more details. This bit is read as 1 on any logical processor that supports the 1-setting of the "unrestricted guest" VM-execution control.
- Bits 8:6 report, as a bitmap, the activity states supported by the implementation:

  — Bit 6 reports (if set) the support for activity state 1 (HLT).

  — Bit 7 reports (if set) the support for activity state 2 (shutdown).

  — Bit 8 reports (if set) the support for activity state 3 (wait-for-SIPI).

If an activity state is not supported, the implementation causes a VM entry to fail if it attempts to establish that activity state. All implementations support VM entry to activity state 0 (active).

- Bits 24:16 indicate the number of CR3-target values supported by the processor. This number is a value between 0 and 256, inclusive (bit 24 is set if and only if bits 23:16 are clear).
- Bits 27:25 is used to compute the recommended maximum number of MSRs that should appear in the VM-exit MSR-store list, the VM-exit MSR-load list, or the VM-entry MSR-load list. Specifically, if the value bits 27:25 of IA32_VMX_MISC is N, then 512 * (N + 1) is the recommended maximum number of MSRs to be included in each list. If the limit is exceeded, undefined processor behavior may result (including a machine check during the VMX transition).
- Bits 63:32 report the 32-bit MSEG revision identifier used by the processor.
- Bits 15:9 and bits 31:28 are reserved and are read as 0.